Advant® OCS

Advant® Fieldbus 100 Interface

AdvaSoft™ for Windows

User’s Guide

Use of **DANGER**, **WARNING**, **CAUTION**, and **NOTE**

This publication includes, **DANGER**, **WARNING**, **CAUTION**, and **NOTE** information where appropriate to point out safety related or other important information.

DANGER	Hazards which could result in severe personal injury or death
WARNING	Hazards which could result in personal injury
CAUTION	Hazards which could result in equipment or property damage
NOTE	Alerts user to pertinent facts and conditions

Although **DANGER** and **WARNING** hazards are related to personal injury, and **CAUTION** hazards are associated with equipment or property damage, it should be understood that operation of damaged equipment could, under certain operational conditions, result in degraded process performance leading to personal injury or death. Therefore, comply fully with all **DANGER**, **WARNING**, and **CAUTION** notices.

TRADEMARKS

Advant, AdvaCommand, AdvaInform, and AdvaBuild are registered trademarks of ABB Asea Brown Boveri Ltd., Switzerland.

AdvaSoft is a trademark of ABB, Asea Brown Boveri Ltd., Switzerland.

SQL*Net is a registered trademark of Oracle Corporation.

Microsoft, MS and MS-DOS are registered trademarks and Windows is a trademark of Microsoft Corporation.

Intellution and SPECIFIX are registered trademarks and THE FIX, FIX MMI, FIX DMACS, FIX BOS, FIX*ation*, and Click and Slick are trademarks of Intellution Inc.

All other brand or product names are trademarks or registered trademarks of their respective companies or organizations.

NOTICE

The information in this document is subject to change without notice and should not be construed as a commitment by ABB Industrial Systems AB. ABB Industrial Systems AB assumes no responsibility for any errors that may appear in this document.

In no event shall ABB Industrial Systems AB be liable for direct, indirect, special, incidental or consequential damages of any nature or kind arising from the use of this document, nor shall ABB Industrial Systems AB be liable for incidental or consequential damages arising from use of any software or hardware described in this document.

This document and parts thereof must not be reproduced or copied without ABB Industrial Systems AB's written permission, and the contents thereof must not be imparted to a third party nor be used for any unauthorized purpose.

The software described in this document is furnished under a license and may be used, copied, or disclosed only in accordance with the terms of such license.

CE MARKING

This product meets the requirements specified in EMC Directive 89/336/EEC and in Low Voltage Directive 72/23/EEC.

Copyright © ABB Industrial Systems AB 1997.

TABLE OF CONTENTS

Chapter 1 - Introduction

1.1 General Information 1-1
1.2 Equipment Requirements 1-3
1.3 Manual Organization 1-4
1.4 Conventions 1-4
1.5 Related Documentation 1-5
1.6 Release History 1-5
1.7 Terminology 1-6
  1.7.1 Abbreviations 1-6
  1.7.2 Definition of Terms 1-7
1.8 Product Overview 1-9
  1.8.1 AF100 Object Groups and handling 1-10
    1.8.1.1 Extended DB Elements 1-10
    1.8.1.2 DAT Based Objects 1-10
  1.8.2 AF100 Communication software overview 1-12
  1.8.3 AF100 functional overview 1-13
  1.8.4 AF100 Object configuration software 1-16
  1.8.5 Advant Fieldbus 100 Interface hardware 1-16
1.9 User Interface 1-17

Chapter 2 - Installation

2.1 Site-Planning Environment 2-1
2.2 Setup 2-1
  2.2.1 CI board strapping and installation 2-2
  2.2.2 TC modem installation 2-4
  2.2.3 Software installation 2-5
2.3 Shutdown Procedures 2-8
2.4 Start-up Procedures 2-8
2.5 Product Verification 2-9

Chapter 3 - Configuration/Application Building

3.1 Design Considerations 3-1
3.2 Capacity and Performance 3-2
  3.2.1 Compatibility 3-3
3.3 Application Start-up 3-3
3.4 Tutorial 3-4
  3.4.1 Creating a small test configuration using Extended DB Elements 3-4
  3.4.2 Creating a small test configuration using DAT Based Objects 3-21

CONTENTS (continued)

3.5 Application Procedures 3-38
 3.5.1 Configuration and application design 3-38
 3.5.2 Extended DB Elements 3-40
  3.5.2.1 Configuration data files using Extended DB Elements.. 3-40
  3.5.2.2 Generating configuration data files 3-40
 3.5.3 DAT Based Objects 3-42
  3.5.3.1 Configuration data files 3-42
  3.5.3.2 Generating configuration data files 3-43
  3.5.3.3 Speeding up the configuration process for multiple station systems 3-45
  3.5.3.4 Changing bus number 3-46
 3.5.4 Configuration of I/O channel for DAT Based Objects 3-46
  3.5.4.1 Using the AF100 Object Configuration software........ 3-48
3.6 Configuration/Application Building Menus 3-51

Chapter 4 - Runtime Operation

4.1 Product Operation 4-1
4.2 Operating Overview 4-3
4.3 Runtime Tutorial 4-3
4.4 Operating Instructions 4-3
 4.4.1 Runtime configuration of the CI525 and the CI526 interface boards..... 4-4
 4.4.2 Hardware and network diagnostics 4-6
 4.4.3 Rescan AdvaBuildDir 4-10
 4.4.4 Resolve 4-11
 4.4.5 Advant Fieldbus 100 Interface version 4-11
4.5 Runtime Operation Menus 4-12

Chapter 5 - Maintenance

5.1 Preventive Maintenance 5-1
5.2 Hardware Indicator 5-1
5.3 Error Messages 5-1
 5.3.1 CI Errors 5-2
5.4 Fault-Finding and User Repair 5-5

Appendix A - Settings in the ADVASOFT.INI File

A.1 General A-1

CONTENTS (continued)

Appendix B - Object Attributes and Commands

B.1 General B-1
B.2 Analog Input (AI) object type B-1
B.3 Analog Output (AO) object type B-4
B.4 Digital Input (DI) object type B-7
B.5 Digital Output (DO) object type B-9
B.6 Memory block elements B-10
B.7 DAT object type B-11
B.8 System Status object type attributes B-12
B.9 Time Tagged Event (EACOLLECT) object type B-13

Appendix C - COIL file structures

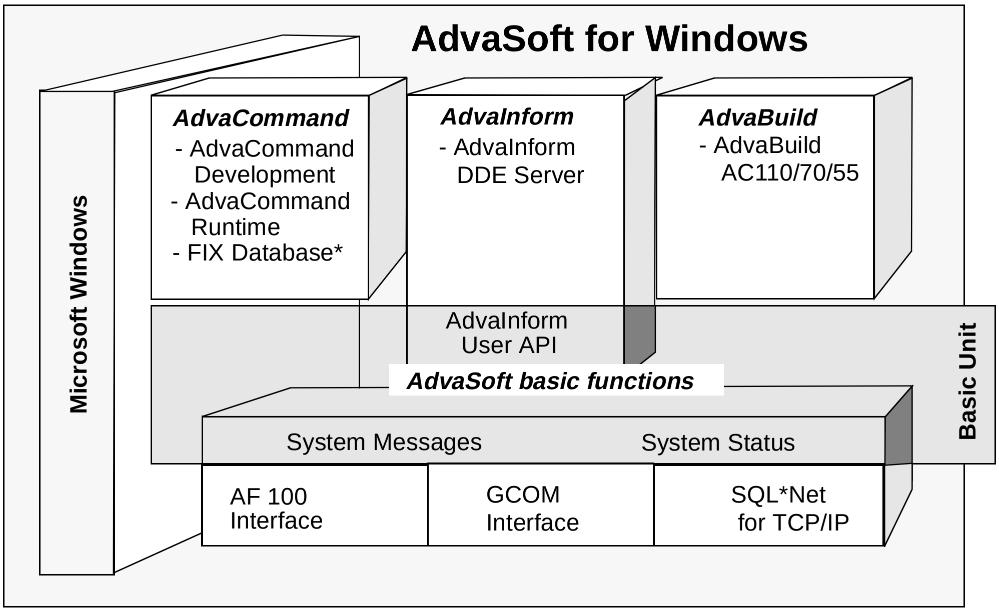
C.1 General C-1
C.2 COIL0000.C00 file C-3
  C.2.1 Structure C-3
  C.2.2 StartLine C-5
  C.2.3 HeadLine C-6
  C.2.4 TerminalLine C-7
  C.2.5 DataSetLine C-9
  C.2.6 DatLine C-11
  C.2.7 EndLine C-12

INDEX

Chapter 1 Introduction

1.1 General Information

This guide describes the Advant Fieldbus 100 (AF100) Interface, which is a combined software and hardware communication option for AdvaSoft for Windows. The Advant Fieldbus 100 Interface is intended to run on a personal computer (PC) in a Microsoft Windows environment and provides communication with stations connected to the Advant Fieldbus 100 network.



* Both versions of AdvaCommand, Development and Runtime, are available with the FIX Database option

Figure 1-1. AdvaSoft for Windows and AF100's place in the program suite

The Advant Fieldbus 100 is a high-performance bus specially designed for real-time applications. It features reliable, cyclic data transfer (using data sets) as well as event-driven background transfer of service data.

If you are a user responsible for installation, configuration, runtime operation and maintenance of the Advant Fieldbus 100 Interface, this manual is intended for you.

To configure and build applications, you require knowledge of configuration and programming with the applications included in the AdvaBuild® for Windows product. For runtime operation of the Advant Fieldbus 100, no previous knowledge is required.

The Advant Fieldbus 100 Interface software provides AdvaSoft for Windows client applications, such as AdvaCommand or custom-built applications, access to process objects in Advant Controllers (AC160, AC110 and AC70), Advant Controller 400 stations and Application Performance Controllers (APC).

Supported functions include:

- Subscription for object attributes and values
- Commands for changing object values
- Handling of time tagged events
- Additional administrative functions
  - Symbol Resolution
  - Time Synchronization
  - Mutual lifesign supervision
  - Application Version Control
  - Configuration of DAT based objects.

You can also use the Advant Fieldbus 100 Interface together with AdvaBuild for Windows for configuration and diagnostics of the Advant Controller 110 via the Advant Fieldbus 100 network.

The figure below shows an example of an Advant Fieldbus 100 Interface configuration with possible redundant cables in the ABB Open Control System (OCS).

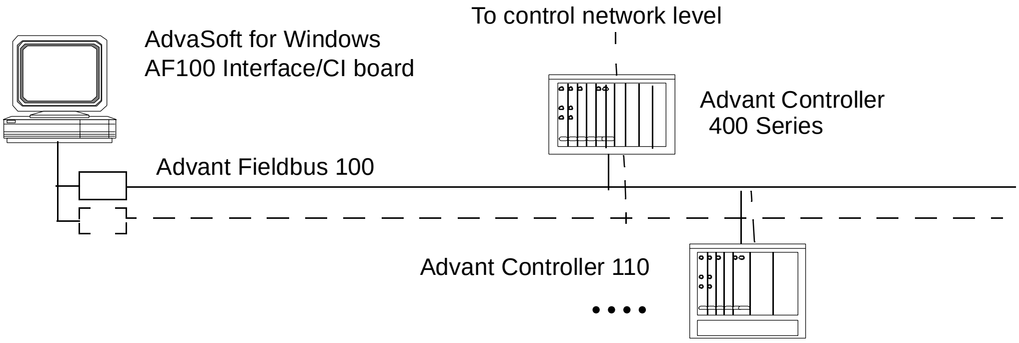


Figure 1-2. Advant Fieldbus 100 Interface in the ABB OCS

1.2 Equipment Requirements

Basic equipment requirements for the Advant Fieldbus 100 Interface are as follows:

Hardware

- PC with 80486/DX2 66MHz or higher CPU
- One 1.44 Mbyte 3.5" diskette drive
- A hard disk with at least 2 Mbyte of available disk space
- 16 MByte of RAM memory
- One free ISA/EISA bus slot for the CI board.

Operating system

- MS-DOS 5.0 or later
- MS-Windows 3.1 or 3.11.

Application software

One or more of the following:

- AdvaCommand for Windows
- AdvaBuild for Windows
- AdvaInform DDE server
- User applications using the AdvaInform User API.

Additional hardware requirements, depending on the application software, are explained in the Equipment Requirements section of the *AdvaSoft for Windows User's Guide*.

1.3 Manual Organization

The figure below shows an overview of the structure of this guide.

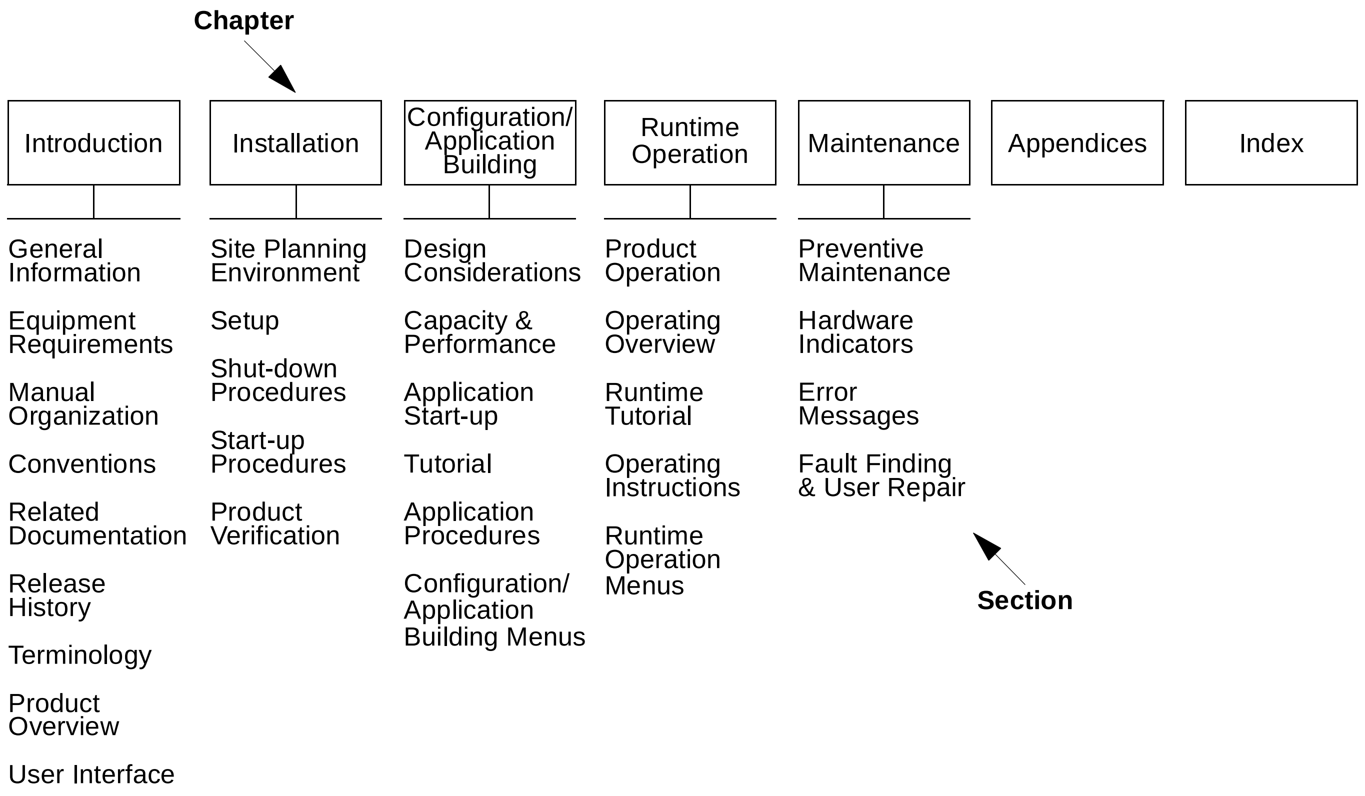


Figure 1-3. Guide structure

1.4 Conventions

The following conventions are used throughout this guide:

Example	**Type of information**
AdvaSoft for Windows User's Guide	References to other books are written in italics.
…<Ctrl-C>…	A dash is used to indicate that you must simultaneously press several keys, Ctrl and C in this example.
…press the OK button…	Push and toggle keys buttons are written in the same way as they appear on the display.
…the File menu…	Initial capital letters are used for the names listed in the menu bar.
…select Exit from the Filemenu…	Initial capital letters are used for the name of menu items.

1.5 Related Documentation

Advant Fieldbus 100
User's Guide

This guide describes how to install and how to use the Advant Fieldbus 100 network. It also provides a technical description of communication boards and modems available.

Function Chart Builder
User's Guide

This guide describes the generation of the DataSet Peripheral and EventSet data prepared for the Advant Fieldbus 100 Interface.

AdvaInform Object Types
Reference Manual

This reference manual describes all objects along with their supported attributes and commands in the ABB automation system.

AdvaSoft for Windows
User's Guide

This guide is the primary document describing the AdvaSoft for Windows product and its available options.

1.6 Release History

Major milestones in the Advant Fieldbus 100 Interface product releases are:

Table 1-1. Advant Fieldbus 100 Interface Release History

Version	Description
1.0	Initial version
1.1	Support for AdvaBuild for Windows
1.2	Support for Extended DB Elements BOIL file handling

1.7 Terminology

1.7.1 Abbreviations

AC
stands for Advant Controller, the common term for the Advant Controller series of controllers used in ABB automation systems, for example, AC 160, AC 110.

AF100
stands for Advant Fieldbus 100.

APB
stands for Application Builder, an application for setting up projects in AdvaBuild for Windows.

APC
stands for Application Performance Controller, a control unit used for motor drives, etc.

AS
stands for Advant Station, the common term for the Advant Station series of operator, engineering and information management stations within ABB automation systems.

BCB
stands for Bus Configuration Builder, an application included in AdvaBuild for Windows for generation of configuration data for the Advant Fieldbus 100 Interface.

BOIL file
stands for Basic Object Information List, a file in ASCII format with configuration data for Extended DB Elements.

COIL file
stands for Configuration Objects Information List, an ASCII file containing configuration data for DAT based objects.

DSP
stands for DataSet Peripheral, an object type to send data via the Advant Fieldbus 100.

EISA
stands for Enhanced ISA. See ISA.

ES
stands for Engineering Station, a common term for stations with functionality for programming and configuration of controllers in the ABB automation system, i.e., AS 100ES.

ESD
stands for Electrostatic Discharge.

EVS
stands for Event Set, the object type for transmitting time tagged events on the Advant Fieldbus 100 network.

FCB
stands for Function Chart Builder, the graphic configuration and programming tool in AdvaBuild for Windows.

HLSC
stands for High Level Series Communication, a communication interface used for communication between AdvaBuild and a controller.

ISA bus
stands for Industrial Standard Architecture. An ISA bus is a standard IBM PC AT compatible internal 16-bit bus.

LED
stands for Light Emitting Diode. LEDs are often used as indicators on control panels.

OS
OS stands for Operator Station, a station with operator interactive functionality.

PC
is an abbreviation for both personal computer and process control.

SDP
SDP stands for the Service Data Protocol used to send messages on the Advant Fieldbus 100 network.

TT_Event
TT_Event stands for time tagged event data transferred on Advant Fieldbus 100 via EVS.

1.7.2 Definition of Terms

Advant OCS
The term refers to one or more of the distributed control systems from ABB Industrial Systems, i.e., ABB Master and ABB MOD 300.

Bus Master
A bus master is a device which administrates the information flow on the Advant Fieldbus 100.

CI Board
A CI board is a Communication Interface Board. There are two types of CI boards, CI525 and CI526. The only difference between them is that the CI526 supports clock slave functionality.

Client Application
A client application is an application which subscribes to objects from the automation system. For example, an application that subscribes to objects via the DDE Server is a DDE client.

Clock Master
A clock master is a device which distributes time on the bus to synchronize clocks in nodes on the Advant Fieldbus 100.

DAT Based Objects
These objects are supported in all versions of AdvaSoft. They offer, compared with the Extended DB Elements, limited object support. DAT based objects are AI, AO, DI and DO. They use COIL files for configuration.

Extended DB Elements
These objects offer complete support for the different object attributes. Extended DB Elements are AIS, AOS, DIS and DOS. They use BOIL files for configuration.

Function chart
A function chart is a diagram for the representation of process control programs. It contains

graphical symbols (rectangles) for PC elements and lines for connections between element terminals.

Object (Process)
Object (Process) is a physical or calculated process object, or a functional unit or type circuit containing all related inputs and outputs. An example of a process object is an analog input including value, limits, status, etc. There are two kinds of objects, Extended DB Elements and DAT based objects.

PC element
PC element is an abbreviation for process control element, which is part of a process control program. PC elements are represented by rectangles in a function chart and by corresponding callnames in a tree diagram.

PC program
PC program is an abbreviation for process control program.

1.8 Product Overview

The Advant Fieldbus 100 Interface contains the hardware and software to enable data exchange between MS-Windows applications and stations on the Advant Fieldbus 100 network.

The selected viewer application uses DDE (Dynamic Data Exchange) to access the data from AdvaInform DDE Server or directly via Object Access Services provided by the AdvaInform User API, see Figure 1-4.

The Advant Fieldbus 100 Interface consists of the following components:

- The AF100 Communication software
- The AF100 Object configuration software
- CI board
- Modem(s).

Below is a conceptual overview of the Advant Fieldbus 100 Interface and related software.

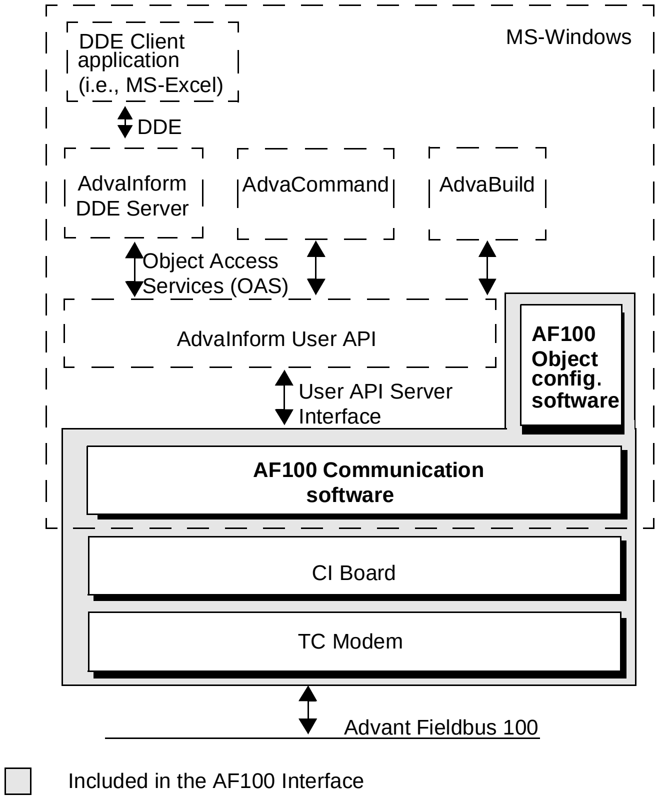

MS-Windows
DDE Client application (i.e., MS-Excel)
DDE
AdvaInform DDE Server
AdvaCommand
AdvaBuild
Object Access Services (OAS)
AdvaInform User API
User API Server Interface
AF100 Object config. software
AF100 Communication software
CI Board
TC Modem
Advant Fieldbus 100
Included in the AF100 Interface

Figure 1-4. Advant Fieldbus 100 Interface software conceptual overview

1.8.1 AF100 Object Groups and handling

The AF100 Interface supports two groups of objects:

- Extended DB Elements
- DAT based objects.

In general, the Extended DB Element group offers more complete object type support with a simplified configuration, as compared with the DAT based object group.

1.8.1.1 Extended DB Elements

The Extended DB Elements provide almost complete support of the object types defined in the *AdvaInform Object Types References* manual. The object information is available through the BOIL files (Basic Object configuration Information List) created with the AdvaBuild application.

The Object Configuration utility is not used for this group of objects.

1.8.1.2 DAT Based Objects

These objects are supported by all versions of the Advant Fieldbus 100 Interface, including versions 1.0 and 1.1. This group is characterized by the fact that the object information is available through COIL files (Configuration Object Information List). The COIL files are created using the AdvaBuild application and the Bus Configuration builder.

The AF100 Object Configuration utility is used for configuration of (static) object attributes not provided from the AF100 Interface network, such as object description, engineering unit, etc., see also Figure 1-5.

From an object (AI, AO, DI or DO) point of view, the mapping can be seen as a combination of DSPs and Event Sets providing required information to the AF 100 Communication handler to implement the full set of the object functions, Figure 1-5.

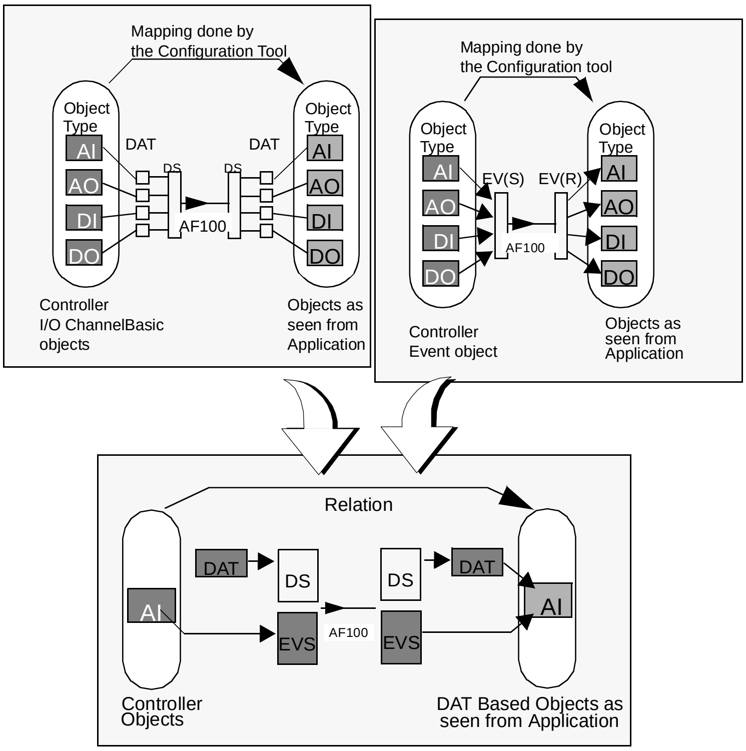


Figure 1-5. The linking between the controller objects and the objects seen from the application using the Object Configuration dialog.

1.8.2 AF100 Communication software overview

The AF100 Interface consists of two parts:

- **The AF100 Communication software,**
  which provides functions for communication with other stations connected to the AF100 network.
- **The AF100 Object Configuration Utility,**
  which enables you to configure **DAT based object** attributes that are not provided from the AF100 network, such as object description, engineering unit, etc.

  For an **Extended DB Element,** all information is taken either from the AF100 network or the BOIL configuration files.

The AF100 communication software handles the communication with the Advant Fieldbus 100 network via the CI board. Configuration data for the AF100 Communication software and CI board is generated by AdvaBuild for Windows.

The AF100 Communication software has two mechanisms for communication:

- **DataSet Peripheral (DSP)**

  DSPs are broadcasted cyclically to all stations on the Advant Fieldbus 100 network. DSP communication is configured and handled by the CI525 Board. The AF100 Communication software has functions to read and write data to these DSPs. The DSPs are used to transfer values for process objects on the Advant FieldBus 100 network.

- **Service Data Protocol (SDP)**

  The SDP is used to transfer messages between applications in different stations on the Advant Fieldbus 100 network. SDP is used to send commands to process objects, to receive time tagged events, and by AdvaBuild for Windows when connecting to the Advant Controller 110 using the AF100 Interface.

1.8.3 AF100 functional overview

The functions of the AF100 Communication software are:

- **Object handling**
  - **Data subscription**

    Values for process objects are received from the Advant Fieldbus 100 network using cyclic DataSet Peripherals (DSP). Alarm Limits, for the **Extended DB Elements** only, are received from the AF100 network using Service Data Protocol.

    Subscription of other object attributes such as object description, engineering unit, etc., is also provided. These attributes are not transferred via the Advant Fieldbus 100 network. The method for retrieving the attributes differs between the **DAT based objects** and the **Extended DB Elements**:

    DAT based object attributes are configured locally in your AdvaSoft for Windows station using the AF100 Object Configuration tool. Only the attribute value is dynamically updated through the network.

    Extended DB Element attributes are obtained from the BOIL files. Status and value are updated dynamically through the network.
  - **Commands**

    Commands can be sent from the AF100 Communication software to change object attributes on the Advant Fieldbus 100 network. Only the attributes **Value** and **Acknowledge** can be changed for DAT based objects in the controller. For Extended DB Elements, the **Value**, **Block**, **Deblock** and **Acknowledge** attributes are changeable in the controller. See also Appendix B, Object Attributes and Commands.
  - **Event handling**

    The AF100 Communication software can be configured to receive time tagged events, via Event Set (EVS), from the Advant Fieldbus 100 network.

    The time tagged events contain object information such as the reason for the event and a time stamp. This information, together with user-defined object information such as object description, engineering unit, etc., can be used by the client application to generate events and alarms.

    For DAT based objects, alarm acknowledgment by the client application can be distributed on the Advant Fieldbus 100 network to other PCs equipped with the Advant Fieldbus 100 Interface. This is done automatically for Extended DB Elements.

See the following table for an overview of the object types and services supported by the Advant Fieldbus 100 Interface object handling functions.

Table 1-2. Object services provided by the Advant Fieldbus 100 Interface

Object Type	Demand	Cyclic	Event	Command
AI	yes	yes	no	yes
AO	yes	yes	no	yes
DI	yes	yes	no	yes
DO	yes	yes	no	yes
DAT	yes	yes	no	yes
SYSTEM_STATUS	yes	yes	no	no
TT_EVENT	no	no	yes	no
MR, MI, MIL and MB	yes	yes	no	yes

For information on all commands and subscription of attributes supported for the different object types, refer to Appendix B, Object Attributes and Commands.

- **Application Version Control**

  This function ensures that on- and off-line changes, made by AdvaBuild in any controller version supporting this functionality, are reflected automatically in the configuration of all AdvaSoft Stations connected to the AF100 network.

- **Time Synchronization**

  This function enables the AF100 user to time synchronize all stations connected to the AF100 network. This functionality depends on which CI board is installed.

- **Two integer byte DAT handling (DAT based objects only)**

  You can increase the bandwidth of the AF100 bus. In doing so, you increase data transport capacity by mapping two integer word DATs, 2 bytes each, into a four byte DAT. You can also scale the object value confined in the integer word DAT object if the value is bigger than 65535. The scaling goes for both Data Subscription and Command sending.
  The scaling factor can be one of 0.01, 0.1, 10, 100 and 1000.

- **AC160 support**

  The CPU handling features are handled internally in the controller and are, therefore, transparent for the AF100 Interface software.

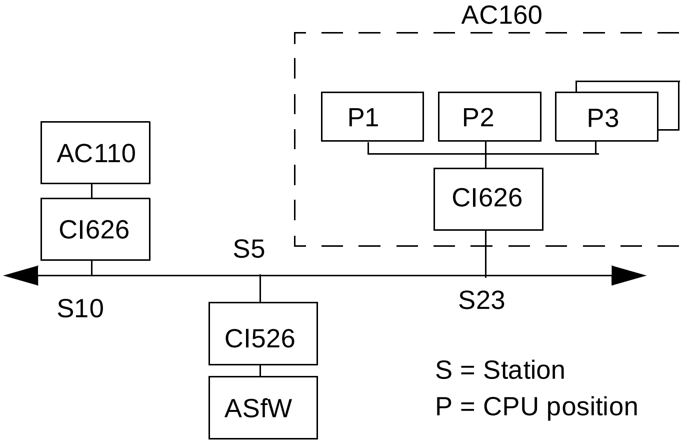


Figure 1-6. Multiprocessing and redundancy

  In Figure 1-6, the CPU position 3 of the AC 160 consists of a redundant CPU pair. Normally, the user will never be concerned about the internal CPU configuration in the controller.

- **AdvaBuild for Windows support**

  The AF100 Communication software provides functions for AdvaBuild for Windows to communicate with stations on the Advant Fieldbus 100 network.

- **Hardware configuration and diagnostics**

  The AF100 Communication software provides functions to view and modify the configuration of the CI525 interface board. Functions to view diagnostics and errors on the CI525 board are also provided.

  The bus lines can be single or redundant. If you use redundant lines, the stations monitor the two lines and automatically select the working line in case of a fault in the bus interface or a broken cable.

  Advant Fieldbus 100 features a distributed bus master scheme. If one or several stations are lost, the bus is not affected and operation continues.

1.8.4 AF100 Object configuration software

Use the AF100 Object configuration software to define object configuration data, for DAT based objects only, which is not provided by the AdvaBuild-produced configuration files.

The following functions are provided:

- **Setup of I/O object type**

  Object values are, by default, treated as DAT objects. With the AF100 Object configuration software, you can map DAT objects to the correct I/O channel type (AI, AO, DI or DO).

- **Setup of object name**

  Objects are, by default, accessed with the DAT object name. You can configure a user-defined object name, for example, the I/O channel name, using this application.

- **Configure object attributes**

  The AF100 Object configuration software provides functions for configuration of object attribute information, such as object description, engineering unit, process section, etc., which cannot be received from the Advant Fieldbus 100 network.

- **Connection to event channels**

  An object must be connected to the correct event channel to properly handle time tagged events generated for the object.

1.8.5 Advant Fieldbus 100 Interface hardware

The Advant Fieldbus 100 Interface hardware consists of the following:

- **CI board**
- **TC modems**

  One or two modems (single or redundant) are provided for coax cable/twisted pair or, alternatively, one modem is provided for single opto cable.

- **Mounting plate**

  A mounting plate is provided to mount the modems and the power supply unit.

- **Power supply unit**

  Power supply is provided for the TC modem(s) mounted on the same mounting plate.

- **Connection cable**

  One or two cables are provided to connect the CI board and the modem(s).

1.9 User Interface

Advant Fieldbus 100 Interface software provides a standard MS-Windows user interface for the two applications included in the product that are to be operated by a user.

The AF100 Communication software and the AF100 Object configuration software are represented by the following icons in the AdvaSoft program group.

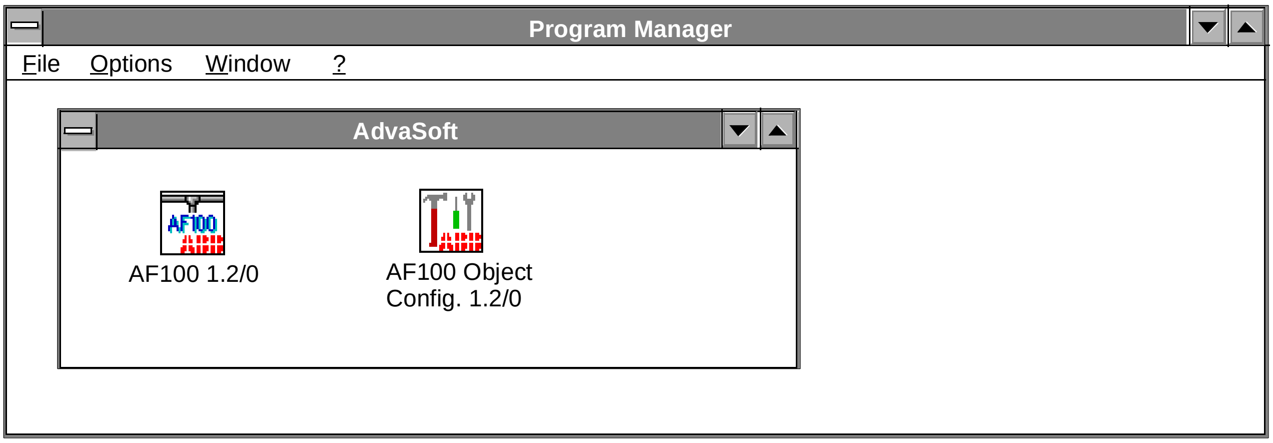


Figure 1-7. Advant Fieldbus 100 software User Interfaces

Chapter 2 Installation

2.1 Site-Planning Environment

When you are site planning, it is important that you check that requirements for environmental conditions (temperature, humidity, chemical), grounding and power supply are met, according to standards for your application. These requirements are stated in the hardware documentation for your PC and in the *Advant Fieldbus 100 User's Manual*.

NOTE

It is important that the PC and the external TC modem(s) be connected to the same grounding point.

Connect the mounting plate as shown in Figure 2-1.

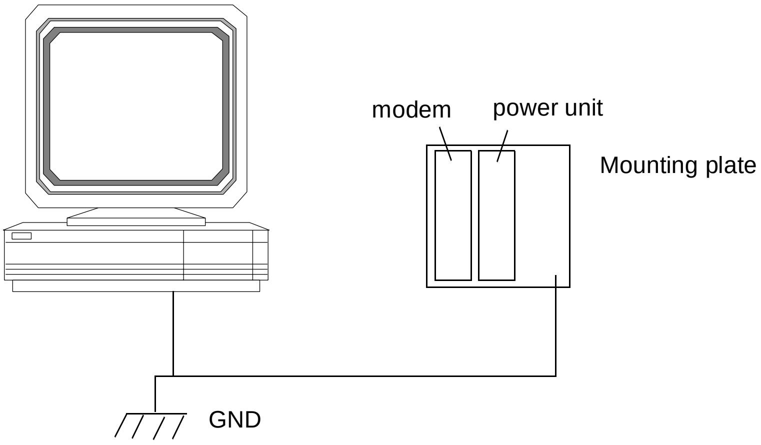


Figure 2-1. PC and TC modem grounding requirements

2.2 Setup

This chapter covers the installation and setup of the Advant Fieldbus 100 Interface, including the following:

- Strapping and installation of the CI board.
- Installation of the modem and the power unit.
- Specific installation steps for the AF100 software.

For a description of how to connect the modem to the AdvantFieldbus 100 network, see the *Advant Fieldbus 100 User's Guide*.

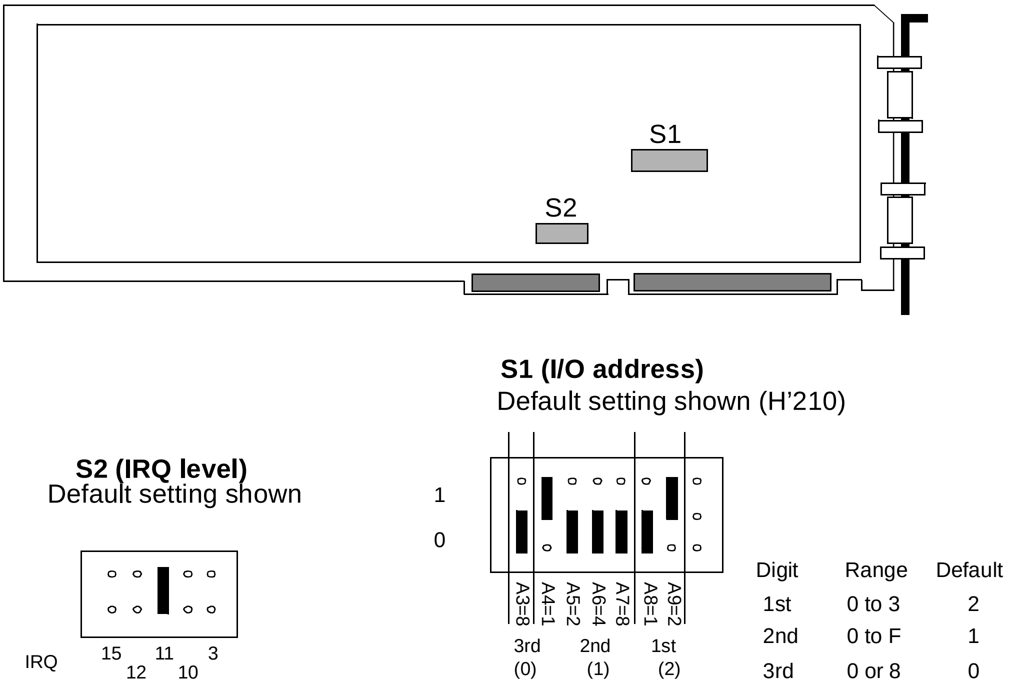
2.2.1 CI board strapping and installation

The CI board has default settings for the I/O address (H'210) and interrupt level (IRQ 11). Before mounting the CI board, check that these default settings are not being used by any other device in your PC.

If you need to change the default settings, follow the instructions below.

CAUTION

To avoid electro-static discharge (ESD) damage to the CI board, we recommend that you exercise preventive maintenance precautions such as using a grounded wrist band and anti-static pad when handling the CI board and other electronic boards. Avoid skin contact with components.



Digit	Range	Default
1st	0 to 3	2
2nd	0 to F	1
3rd	0 or 8	0

Figure 2-2. CI board strapping

IRQ level selection

Set IRQ by placing the strap on the required IRQ according to the diagram in Figure 2-2.

I/O address selection

The S1 connection block includes bridges for setting binary bits A3 to A9 to 0 or 1. The default settings are 2, 1 and 0, respectively (all digits are in Hex), and are shown in Figure 2-2.

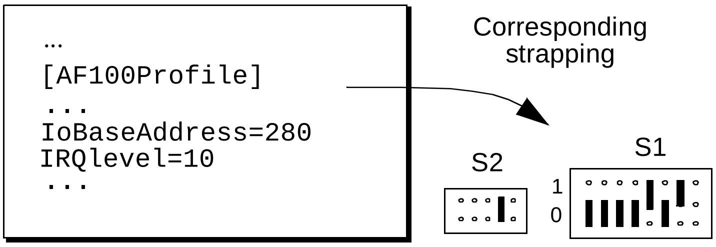
- A3 comprises the most significant bit (8) in the least significant digit of the I/O address. Bits 0, 1 and 2 are preset to 0 and cannot be modified.
- A4 to A7 comprise digit 2, where A4 is the least significant bit and A7 is the most significant bit.
- A8 and A9 comprise the first digit, where A8 = 1 and A9 = 2.

Modifications in the ADVASOFT.INI file

Changes of I/O address or IRQ level must be updated in the ADVASOFT.INI file (residing in the MS-Windows directory). Insert the following lines in the AF100Profile section:

```
IoBaseAddress=<I/O address in hex format>
IRQlevel=<interrupt level>
```

An example of the AF100Profile section in the ADVASOFT.INI file follows:



```
...
[AF100Profile]
...
IoBaseAddress=280
IRQlevel=10
...
```

Figure 2-3. Example of CI board setup in the ADVASOFT.INI file

For more information on settings in the ADVASOFT.INI file, refer to Appendix A, Settings in the ADVASOFT.INI File.

CI board installation

When you have finished strapping the CI board, follow the mounting instructions below and the instructions in your PC manual regarding how to add cards on the ISA/EISA bus.

WARNING

Cut main power supply and read the instructions supplied with your PC's documentation before you start to disassemble.

Install the CI board as follows:

1. Disconnect the PC's main power supply.
2. Open up the PC's chassis and locate an empty ISA bus slot.
3. Remove a spare board slot cover.
4. Plug the CI board firmly into place and secure with fixing screw.
5. Close the PC's chassis and reconnect the main power supply.

2.2.2 TC modem installation

Connect the connection cable to the CI board, the modem(s) and the power supply unit, as shown in Figure 2-4.

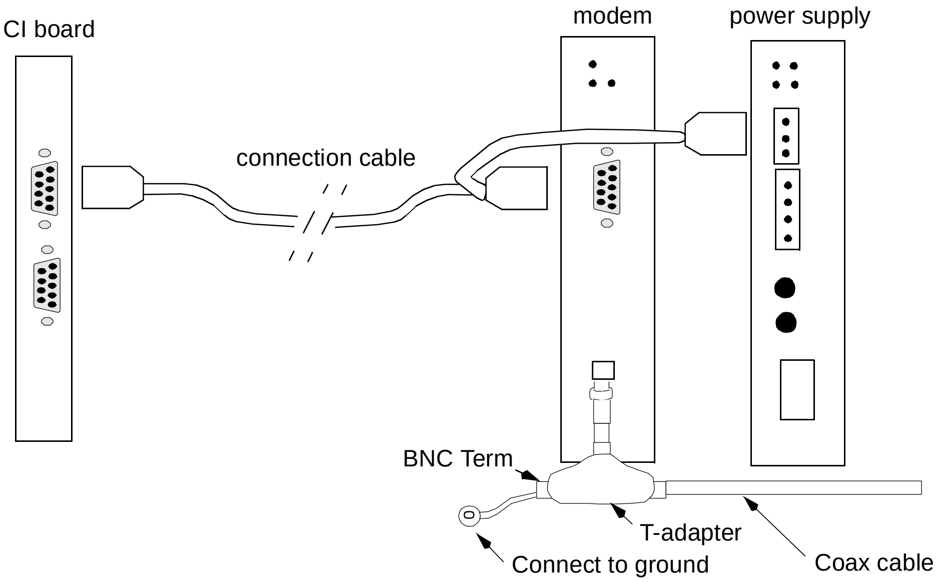


Figure 2-4. Modem installation

The modem is connected to the Advant Fieldbus 100 network with a BNC T-adapter. Refer to the *Advant Fieldbus 100 User's Guide* for more information.

NOTE

It is important that both ends of the Advant Fieldbus 100 network cable be equipped with 75 ohm terminations. **One** of the terminations must also be connected to ground.

2.2.3 Software installation

The software should already be installed, in accordance with the installation description in the *AdvaSoft for Windows User’s Guide*. There are some installation steps, however, that are specific to the AF100 software, which are not covered in the *AdvaSoft for Windows User’s Guide*.

NOTE

It is recommended that you install the AdvaBuild option (if it is a part of the system) before installing the AF100 option so the shared directory structure will be automatically created.

During the installation of the Advant Fieldbus 100, which is done in the AdvaSoft for Windows installation program, the following dialog is displayed:

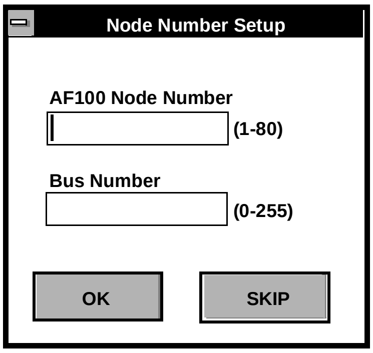


Figure 2-5. Node Number Setup dialog box

1. Assign a node number and bus number for the local node.

   If AdvaSoft for Windows is already installed and station and bus numbers are defined in the ADVASOFT.INI file, this station number is the default in the AF 100 Station Number field. If no previous AdvaSoft installation exists, this field is empty. You have these three options:

   – **Enter a new station number.**
   Enter the new number after you delete any default number. Note that the station number has to be within the range of 1 to 80 and cannot be used by any other station on the bus.

   – **Accept the default station number.**

   – **Leave the field empty.**
   AdvaSoft’s installation program does not take any action concerning station number.

   Fill in the number for the bus to use for the local node in the Bus Number text box. The bus number is used for grouping together stations that are related to each other in some way. They may be related because they are in the same process section or project. The bus number does not refer to a physical bus. The same fieldbus is always used.

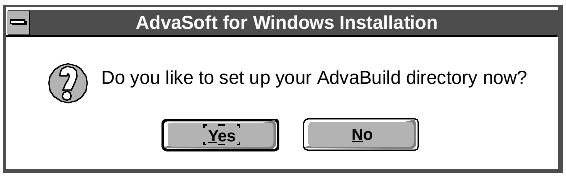
You can also leave this text box empty.

When you have finished, click the OK button to confirm the information entered.

This information is written to the ADVASOFT.INI file.

If you click the SKIP button, the installation program proceeds to the next step without taking any actions regarding this dialog. After installation is completed, edit the ADVASOFT.INI file manually to fill in any information that was left out.

A new message box is displayed



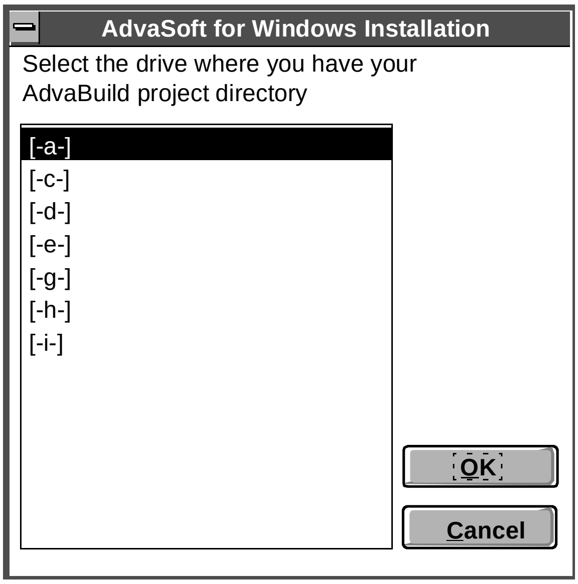
Do you like to set up your AdvaBuild directory now?

Figure 2-6. Dialog for adding nodes

2. Click Yes or No.

   If you click **No**: the installation program does not scan for any …\NODES directories on local or network drives and proceeds to step 6.

   If you click **Yes**: the following dialog is displayed.



Select the drive where you have your AdvaBuild project directory

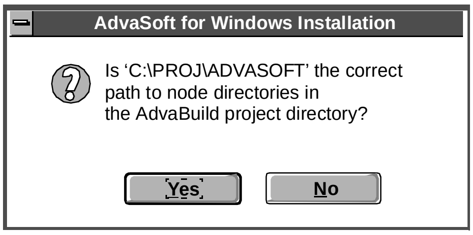
Figure 2-7. Network drives dialog

The AdvaBuild project directory can reside on the AdvaBuild station if the AdvaBuild and the AdvaSoft stations are connected to the same local network.

3. Choose the hard drive that holds the AdvaBuild project directory.

   The installation program scans the selected network drive for the …\NODES directory and writes the search path into the ADVASOFT.INI file.

   The following dialog is presented for each …\NODES directory found:


AdvaSoft for Windows Installation

Is 'C:\PROJ\ADVASOFT' the correct path to node directories in the AdvaBuild project directory?

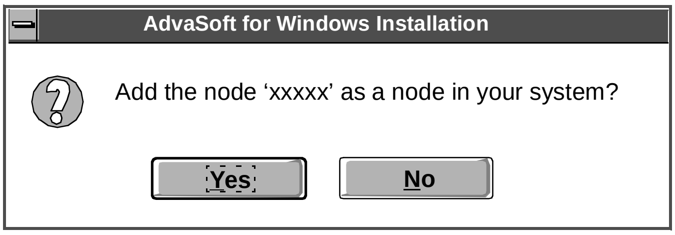
Figure 2-8. Dialog for confirming the node path

4. Select your AdvaBuild project directory.

   The search paths for every found ...\NODES directory are presented. Click the Yes button to select one.

   If you click the No button, the next path is presented in the dialog.

   When you click the Yes button, the installation program scans the subdirectories for system nodes. A new dialog is displayed for each found node.

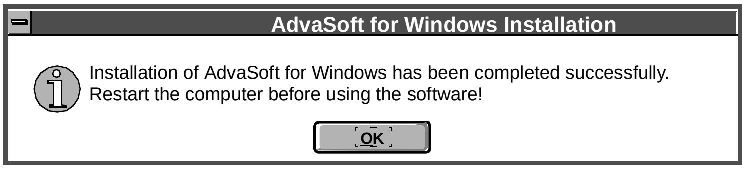

AdvaSoft for Windows Installation

Add the node 'xxxxx' as a node in your system?

Figure 2-9. Dialog asking about adding a node

5. Click the Yes button to confirm if you want the presented node added to the system.

   The following message is displayed when the node configuration is done.



AdvaSoft for Windows Installation

Installation of AdvaSoft for Windows has been completed successfully.
Restart the computer before using the software!

Figure 2-10. Completion message

6. Click the OK button.

   The installation creates the new AF100 icons (or updates those already existing) in the AdvaSoft program group.

2.3 Shutdown Procedures

Normal shutdown

The normal procedure to shut down the Advant Fieldbus 100 Interface software is to stop the client application using AdvaSoft for Windows, for example, AdvaCommand for Windows. The CI board goes to a passive state. The CI board also goes to a passive state if the AF100 Interface lifesign stops or the **Disconnect** button is pressed in the configuration dialog.

Emergency shutdown

If the software is not responding or you are unable to shut down the Advant Fieldbus 100 Interface software, exit Microsoft Windows. If this is not successful, **power off** the PC.

2.4 Start-up Procedures

The Advant Fieldbus 100 Interface starts when you start the client application using AdvaSoft for Windows. Alternatively, you can start the Advant Fieldbus 100 Interface by double-clicking the program icon from the Microsoft Windows program manager.

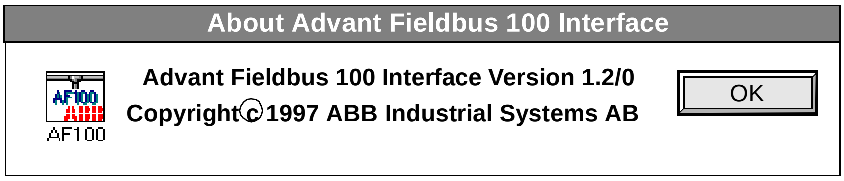
NOTE

Before you can start the Advant Fieldbus 100 Interface, you must define the station number for the CI board. The station number is defined during installation, in the configuration files generated by AdvaBuild for Windows or by manual insertion of the number into the ADVASOFT.INI file. See Appendix A, Settings in the ADVASOFT.INI File.

2.5 Product Verification

To verify installation and contact with the Advant Fieldbus 100 network, do the following:

1. Check that the two icons, AF100 1.2/0 and AF100 Object Config. 1.2/0, for the Advant Fieldbus 100 Interface software are created in the AdvaSoft program group (refer to Figure 1-7).
2. Check that the green P (power) LED on the modem(s) is lighted.
3. Check that the yellow RX (receive) LED on the modem(s) is lighted (if other stations on the Advant Fieldbus 100 network are up and running).
4. Start up the Advant Fieldbus 100 Interface program and choose the **About...** item from the **Control** menu to verify the program version. See Figure 2-11.



Advant Fieldbus 100 Interface Version 1.2/0

Copyright © 1997 ABB Industrial Systems AB

Figure 2-11. Advant Fieldbus 100 Interface Product verification

When the Advant Fieldbus 100 Interface software is started, do the following:

1. Check that the yellow TX (transmit) LED on the modem starts to blink.
2. Check the System Messages application to ensure that no errors are reported from the Advant Fieldbus 100 Interface software.
3. Choose the **Status...** item from the Advant Fieldbus 100 **Control** menu and verify that the board is operational and that no errors are detected.

Chapter 3 Configuration/Application Building

3.1 Design Considerations

Some user configuration is necessary before the Advant Fieldbus 100 Interface can begin to exchange data with stations on the Advant Fieldbus 100 network. Note that during the configuration phase, the two types of objects, **Extended DB Elements** and **DAT based objects**, are handled in different ways.

- **Creation of configuration data files**

  The configuration files are generated with AdvaBuild for Windows. For the working method for creating these files, see Section 3.5.2, Extended DB Elements and Section 3.5.3, DAT Based Objects. For a basic tutorial, see Section 3.4, Tutorial.

  The configuration files contain setup information for the CI board and DAT object information for the Advant Fieldbus 100 Interface to facilitate proper communication with other stations on the Advant Fieldbus 100 network. Two kinds of configuration files exist, BOIL and COIL files. BOIL files are for handling the Extended DB Elements objects and COIL files are for handling DAT based objects.

- **Mapping of I/O channel for DAT based objects**

  The Advant Fieldbus 100 Object Configuration program is used to map the correct I/O channel type to a DAT based object defined in the configuration data files. You can also do configuration of object attributes, such as object description, engineering unit, etc., and connection to time tagged events with this utility. Refer to Section 3.5.4.1, Using the AF100 Object Configuration software for more information.

  No configuration of Extended DB Elements is necessary since all object information is present in the BOIL configuration files. Furthermore, Extended DB Elements are not accessible or changeable from the AF100 Object Configuration software.

- **Updating the ADVASOFT.INI file parameters**

  Update the ADVASOFT.INI file in the following case:

  - If acknowledgment of alarms for DAT based objects is to be distributed to other PC stations equipped with the Advant Fieldbus 100 Interface.

  Refer to Appendix A, Settings in the ADVASOFT.INI File for more information.

NOTE

It is strongly recommended that you upgrade your system if AdvaSoft stations, version 1.1/0, or Advant Controllers 110, version 2.0/4, is present in your system. In the case of AdvaSoft for Windows, upgrade to version 1.1/1 or later. In the case of Advant Controller 110, upgrade to version 2.0/5 or later.
Some Advant Fieldbus 100 (version 1.2) commands (when you push the Detail button in the Status dialog) may result in an AdvaSoft or controller stall.

3.2 Capacity and Performance

The capacity and performance of the Advant Fieldbus 100 Interface are described in this section. Performance depends on the client application program and on the MS-Windows system's load. Refer to the *AdvaSoft for Windows User's Guide* and the *AdvaCommand Development, AdvaCommand Runtime User's Guide* for more information.

The capacity of the AF100 Communication handler is indicated in the table below:

Table 3-1. Capacity of the AF100 Communication handler

Description	Data
Total number of objects	30000
Subscription times	100ms - 9h

The capacity of the CI board is indicated in the table below. To calculate the Advant Fieldbus 100 bus load, refer to the *Advant Fieldbus 100 User's Guide*.

Table 3-2. Advant Fieldbus 100 Interface Capacity

Description	Data
Total no. of DataSet Peripherals	4000
Cycle times depending on configuration in AC110	1, 2, 4,... 4096 ms
Cycle times depending on configuration in AC400	32, 64,... 4096 ms

The Advant Fieldbus 100 runs a data rate of 1.5 MBits/sec. Some of the information is, however, used for preambles, cyclic redundancy checks and protocol overhead; as a result, the net data rate is somewhat less.

Table 3-3. Data set transfer time

Data set size		Transfer time (ms)
1	DAT element	0.008
2	DAT elements	0.108
3-4	DAT elements	0.156
5-8	DAT elements	0.252

The transfer times for data sets depends on the number of attached DAT elements (see Table 3-3).

The bus load depends on the data set configuration and can be calculated using the following formula:

$$BusLoad = \sum\left(\frac{Nbr \cdot Ttr}{cT}\right) \cdot 100$$

This formula provides the bus load as a percentage, where the sum is taken over all different sending data sets and:

Nbr = number of data sets (of the same data size and *cT*)
Ttr = transfer time in ms (from Table 3-3)
cT = desired cycle time in ms (1, 2, 4, 8, 16…4096).

Because of the organization of the scan table (and all operating principles of the communication interface firmware) some constraints are imposed on Advant Fieldbus 100 transmissions.

To guarantee that message transfer is possible, at least 25 percent of the time slots are reserved for message transfer. The rest of the time slots may be used freely for data-set communication.

3.2.1 Compatibility

The table below shows compatibility for program versions, hardware and the supported hardware and software options.

Table 3-4. Supported options for different controllers

Options	**Controller/Controller version**						
	AC70 ver. 1.0	**AC110 ver. 2.0**	**AC110 ver. 2.1**	**AC160 ver. 1.0**	**AC400 ver. 1.0**	**APC ver. 1.0**	**APC ver. 2.0**
BOIL handling	No	No	Yes	Yes	No	No	No
COIL handling	Yes	Yes	Yes	Yes	No(1)	No	Yes
SDP	Yes	Yes	Yes	Yes	No	No	Yes
DSP	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Event handling	Yes	Yes	Yes	Yes	No(2)	Yes	Yes
AF100/Optical Bus	No	Yes	Yes	Yes	Yes	No	No
AF100/COAX. Bus	No	Yes	Yes	Yes	Yes	Yes	Yes
AF100/Twisted Pair	Yes	Yes	Yes	Yes	Yes	No	No

(1) The COIL file residing in the controller requires manual editing.
(2) No event distribution when using the AF100 Interface.

3.3 Application Start-up

Start the AF100 Object Configuration program from the icon in the AdvaSoft program group.

3.4 Tutorial

3.4.1 Creating a small test configuration using Extended DB Elements

By following the steps in this tutorial, you create a small basic application that you can use to test whether the AF100 Interface is up and running after the installation procedure is completed. To carry out this tutorial, the following applications and hardware must be installed and working:

- AdvaSoft for Windows, Version 1.3
- AF100 Interface, Version 1.2
- AdvaBuild for Windows, Version 1.3
- AdvaCommand for Windows, Version 1.2
- One AC110 controller, Version 2.1 with, at least, the Extended PC 3.0 (OPT 1) and Events 3.0 (OPT3) options installed.

This tutorial assumes that this is a new installation with no previous node or network configuration. During the tutorial, a "procedure bar" is placed at strategic positions, indicating where in the procedure the tutorial is at a given moment.

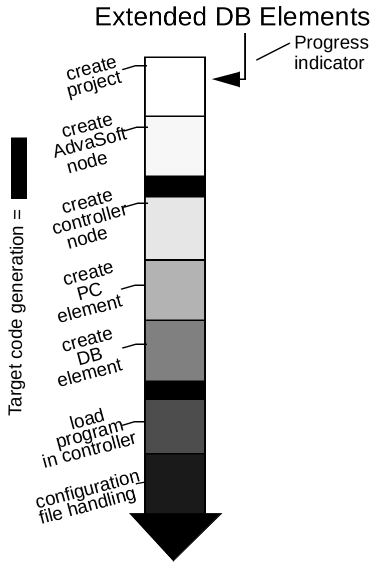
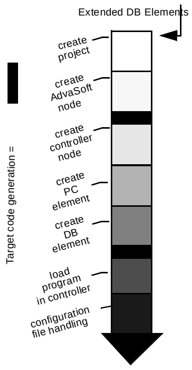


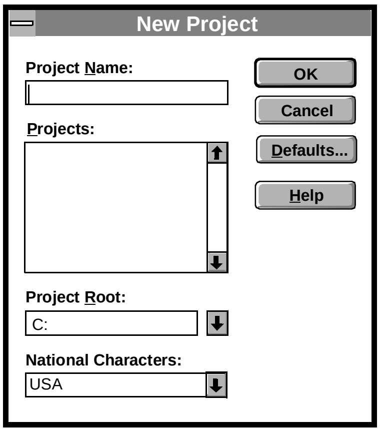
Figure 3-1. Procedure bar used for Extended DB Elements



Creating a new project

1. Start the AdvaBuild Application Builder program.
2. Select the New/New Project item in the File menu.

   The following dialog is displayed



Project Name:

Project Root: C:

National Characters: USA

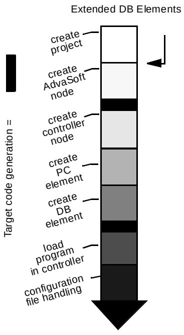
Figure 3-2. New Project dialog

3. In the Project Name field, fill in following information.

   Set Project Name to AdvaSoft (if a project name other than the project name stated during the installation process is used, then you must manually edit the *Advabuilddir* item in the ADVASOFT.INI file).

   Set Project Root to C: (or as appropriate).
4. Click OK.

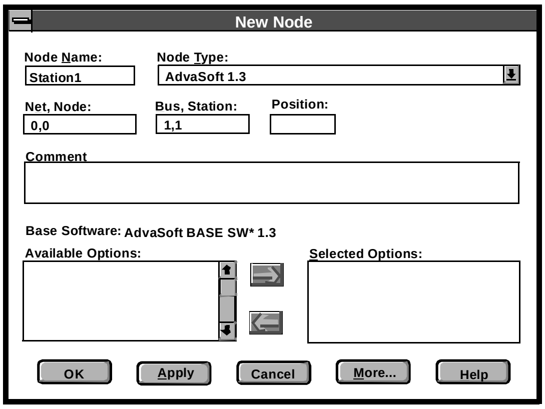
   Three windows are displayed:
   - The Node List window
   - The Circuit List window
   - The Type Circuit List window.



Creating a new AdvaSoft station node

5. Select the New Section/New Node item in the File menu.

   The following dialog is displayed:



Node Name: Station1
Node Type: AdvaSoft 1.3
Net, Node: 0,0
Bus, Station: 1,1
Position:
Comment
Base Software: AdvaSoft BASE SW* 1.3
Available Options:
Selected Options:

Figure 3-3. New Node dialog

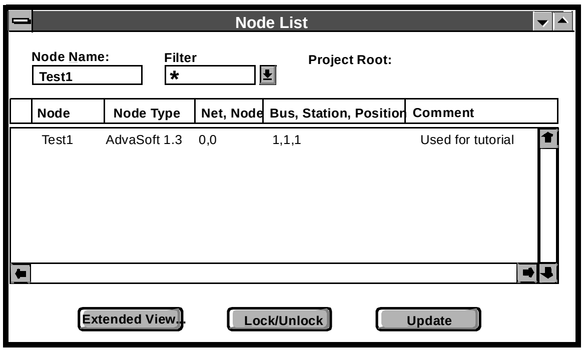
6. In the New Node dialog, fill in following information:
   - Set Node name as appropriate (Station1).
   - Set Node type to AdvaSoft 1.3.
   - Set Net, Node to 0,0.
   - Set Bus, Station as appropriate (1,1).

NOTE

It is important to remember the setting for the bus number when you are configuring the controller node.

7. Click OK.

The Node List dialog is updated:



Node Name: Test1

Filter: *

Project Root:

Node	Node Type	Net, Node	Bus, Station, Position	Comment
Test1	AdvaSoft 1.3	0,0	1,1,1	Used for tutorial

Figure 3-4. Node List dialog

8. Select the new AdvaSoft station node and choose the Function Chart Builder item from the Tools menu.

   The Function Chart Builder application is started. During the start-up sequence, you are asked for the page size used for the chart diagram (for example A4, English landscape). When the start-up sequence is complete, a dialog with the program elements is displayed:

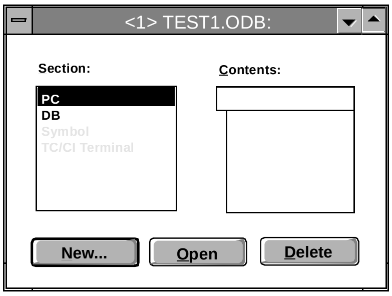
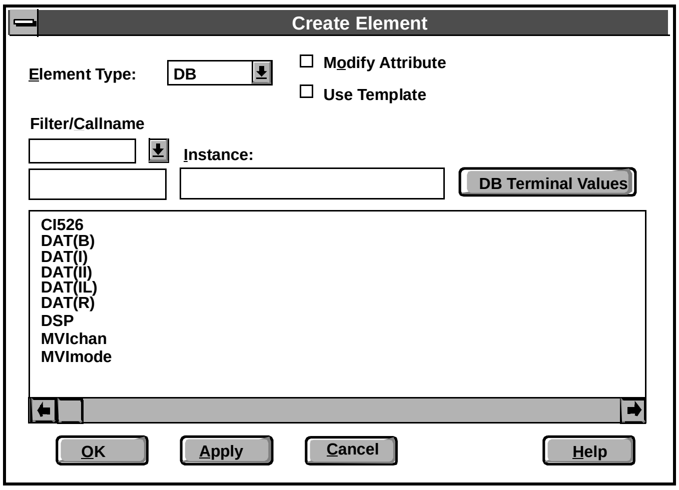


Figure 3-5. Initial dialog in the Function Chart Builder application

9. Choose New Section /DB item in the File menu.

The Create Element dialog is displayed



Create Element

Element Type: DB

☐ Modify Attribute

☐ Use Template

Filter/Callname

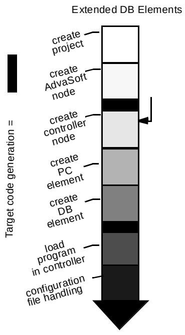
Instance:

CI526
DAT(B)
DAT(I)
DAT(II)
DAT(IL)
DAT(R)
DSP
MVIchan
MVImode

Figure 3-6. Create Element dialog, DB element version

10. In the list box, mark the CI526 (or higher) item and click the DB Terminal Values button.
11. Enter the station and bus number (1,1) and press OK.
12. Click the OK button and then click the OK button in the Create Element dialog.
13. Save the new node by choosing the Save item from the File menu.
14. Choose the Generate Target Code... item from the File menu.

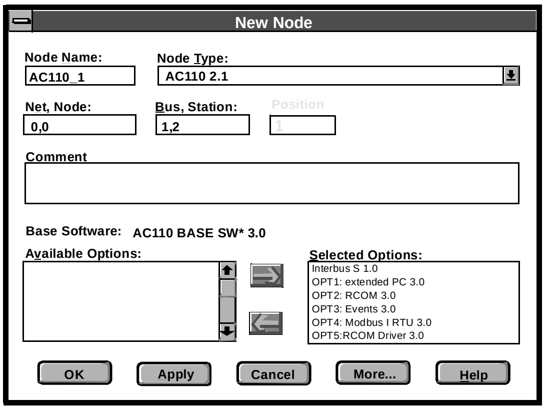
    The target code for the AdvaSoft station is created. Brief information about the compilation procedure is displayed.
15. Exit the Function Chart Builder application.



Creating a new controller node

16. Activate AdvaBuild Application Builder.
17. Select the New/New Node item from the File menu.

    The following dialog is displayed (all fields are empty as default):



Node Name: AC110_1

Node Type: AC110 2.1

Net, Node: 0,0

Bus, Station: 1,2

Position 1

Comment

Base Software: AC110 BASE SW* 3.0

Available Options:

Selected Options:
Interbus S 1.0
OPT1: extended PC 3.0
OPT2: RCOM 3.0
OPT3: Events 3.0
OPT4: Modbus I RTU 3.0
OPT5:RCOM Driver 3.0

Figure 3-7. An example of a filled-in New Node dialog

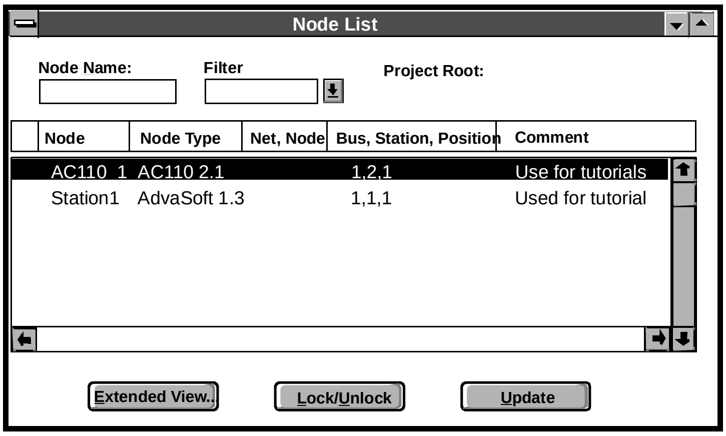
18. In the New Node dialog, fill in following information.
    - Set Node Name as appropriate (AC110_1).
    - Set Node type to AC110 2.1.
    - Set Net, Node to 0,0.
    - Set Bus, Station as appropriate (1,2).

NOTE

The setting for the bus number must be the same as the setting used for the AdvaSoft station with which the controller is to communicate (see earlier step in the tutorial). The station number must correspond with the station number setting in the controller.

19. Select all available options.
20. Click OK.

The Node List dialog is displayed with the newly created controller node listed among the other items in the list box.

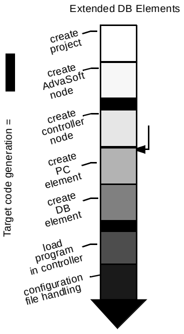


Node	Node Type	Net, Node	Bus, Station, Position	Comment
AC110 1	AC110 2.1		1,2,1	Use for tutorials
Station1	AdvaSoft 1.3		1,1,1	Used for tutorial

Figure 3-8. Nodes created in the project

21. Select the new controller node and choose the Function Chart Builder item from the Tools menu.

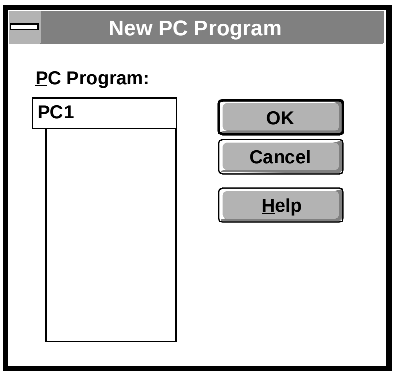
    The Function Chart Builder application is started. An initial dialog is displayed which asks for page size. Choose an appropriate page size.



Building the controller application

22. Activate the Function Chart Builder application.
23. Choose New/PC from the File menu.

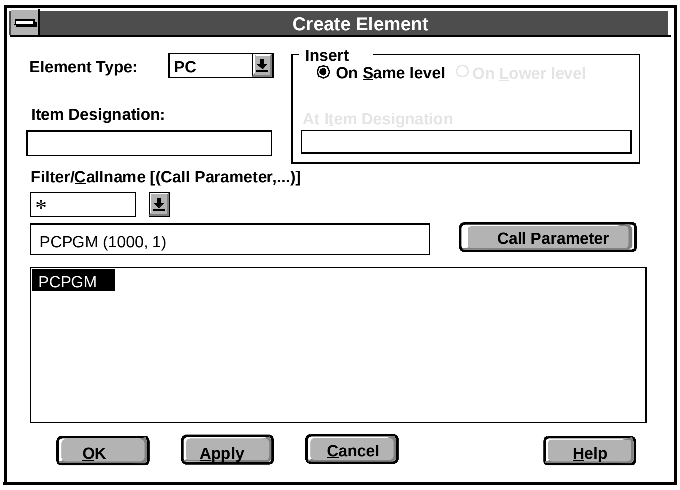
    The New PC Program dialog is displayed:



PC Program: PC1

Figure 3-9. New PC Program dialog

24. Fill in the name of your new PC program (PC1) and click OK
25. The Create Element dialog is displayed:



Element Type: PC

Insert ● On Same level ○ On Lower level

Item Designation:

At Item Designation

Filter/Callname [(Call Parameter,...)]

*

PCPGM (1000, 1)

PCPGM

Figure 3-10. Create Element dialog

26. Click on the PCPGM item in the list box.
27. Click the Call Parameter button.
28. Fill in the cyclicity time, for example, 100 ms, in the displayed dialog and click OK.

    You return to the Create Elements dialog.
29. Click the Apply button.

    A chart showing the items created as a result of the steps above is displayed.

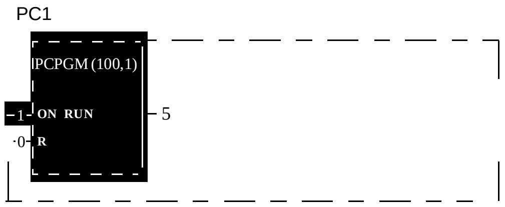
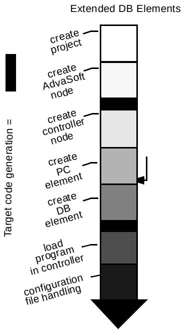
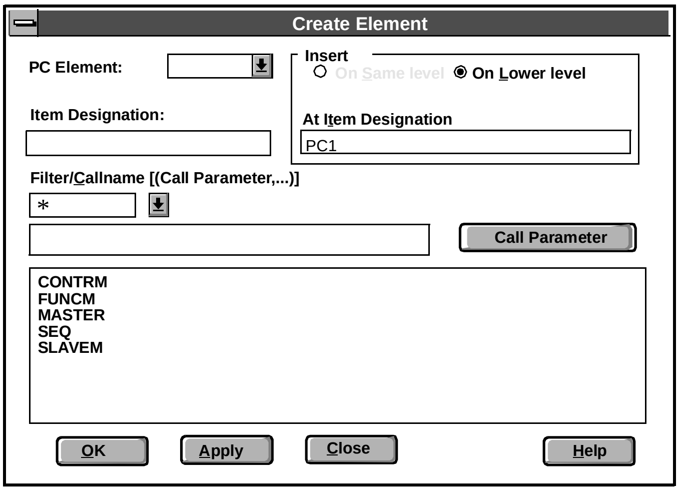


Figure 3-11. Example of the chart displayed after generating target code for a PC program



The Create Element dialog is updated with a new set of parameters in the list box:



PC Element:

Insert: ○ On Same level ● On Lower level

Item Designation:

At Item Designation: PC1

Filter/Callname [(Call Parameter,...)]: *

CONTRM
FUNCM
MASTER
SEQ
SLAVEM

Figure 3-12. Create Element dialog, second version

30. Click the CONTRM item in the list box.
31. Click the Call Parameter button.
32. Fill in the cyclicity time, for example, 50 ms, in the displayed dialog and click OK.

    You return to the Create Elements dialog.
33. Click the Apply button.

    The chart showing the configured program elements is updated with the CONTRM item.
34. Click the Close button in the Create Elements dialog.
35. In the Function Chart Builder application, activate the Chart window (PC Section - PC1) and click on the 1 connection, belonging to the PCPGM item, to mark it.
36. Choose the Connect item from the PC -Terminal menu.
37. Fill in the following in the To section of the displayed dialog:

    D=1
    and click the Apply/Next button.
    Fill in *D=0*
    click the Apply/Next button
    Fill in *D=1*
    click the Apply/Next button
    Fill in *D=0*
    click the Apply/Next button
    Fill in *D=0*
    click the OK button.

After this procedure, the chart should look like this:

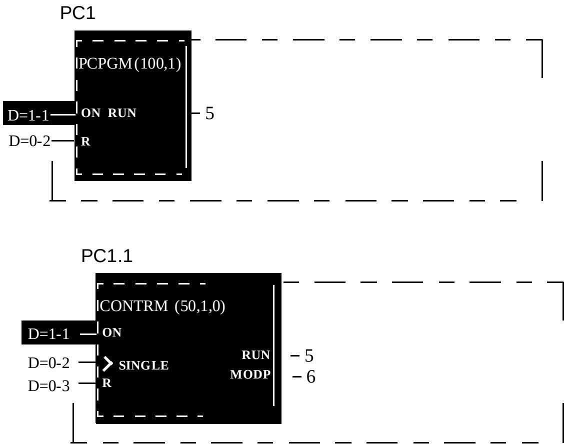
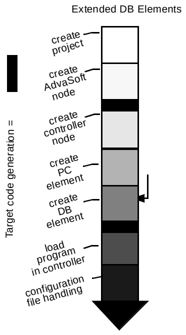
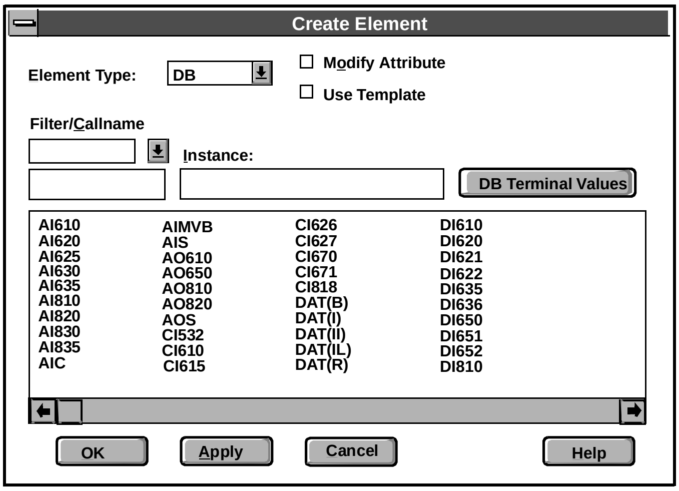


Figure 3-13. Chart after the connection procedure



38. Choose New Section/DB from the File menu.

    The Create Element dialog is displayed:

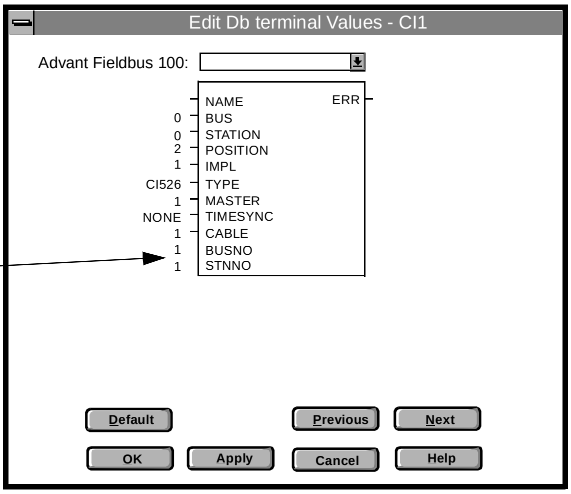


Element Type: DB

Figure 3-14. Create Element dialog, DB element version

39. In the list box, mark the CI626 (or higher) item and click the DB Terminal Values button.

The Edit Db terminal Values - CI1 dialog is displayed.



Advant Fieldbus 100:

Value	Terminal
	NAME
0	BUS
0	STATION
2	POSITION
1	IMPL
CI526	TYPE
1	MASTER
NONE	TIMESYNC
1	CABLE
1	BUSNO
1	STNNO

Note that the default bus number is zero. In this tutorial we will set the bus number to one.

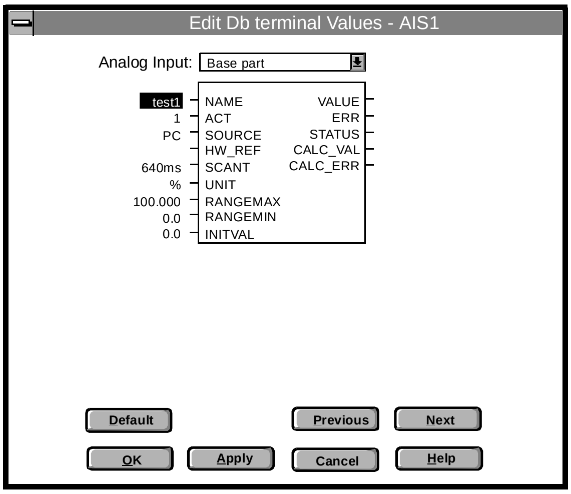
Figure 3-15. Edit Terminal Values dialog

40. Fill in following information:

    Set the bus and station numbers.

41. In the list box, mark the PM632 (or higher) item and click the Apply button.
42. In the list box, mark (click on) the AIS item and--this is important--click the DB Terminal Value button.

    The Edit Db terminal Values - AIS1 dialog is displayed.



Analog Input: Base part

Value	Terminal
test1	NAME
1	ACT
PC	SOURCE
	HW_REF
640ms	SCANT
%	UNIT
100.000	RANGEMAX
0.0	RANGEMIN
0.0	INITVAL

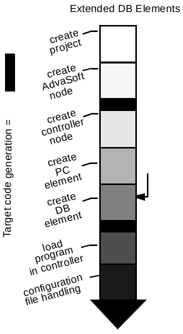
Figure 3-16. Edit Terminal Values dialog

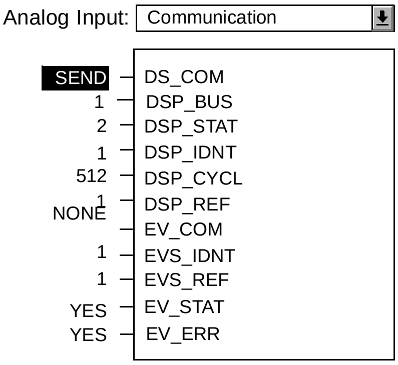
43. Fill in following information:

    Set the Name item to an appropriate object name (test1).

44. Choose the Communication item from the Analog Input drop-down list.

    The parameter listing is updated with communication specific items:





Analog Input: Communication

SEND	DS_COM
1	DSP_BUS
2	DSP_STAT
1	DSP_IDNT
512	DSP_CYCL
1	DSP_REF
NONE	EV_COM
1	EVS_IDNT
1	EVS_REF
YES	EV_STAT
YES	EV_ERR

Figure 3-17. Edit Terminal Values dialog, second version

NOTE

The DSP_STAT and DSP_IDNT parameters explicitly identify a certain DataSet. It is imperative that two DataSets Peripheral with identical DSP_STAT and DSP_IDNT settings not exist in the system. No checks are made for identical DataSets in other controllers when a new one is created. It is recommended that you run the BCB application to avoid identical DataSets in your system.

45. Fill in following information:

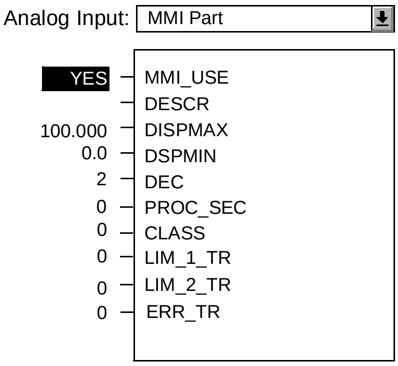
    Set the DS_COM item to send.

    Set the DSP_STAT item to the actual station number (2).

46. Click the Apply button.

47. Choose the MMI Part item from the Analog Input drop-down list.

The parameter listing is updated with MMI specific items:



Analog Input: MMI Part

YES	MMI_USE
	DESCR
100.000	DISPMAX
0.0	DSPMIN
2	DEC
0	PROC_SEC
0	CLASS
0	LIM_1_TR
0	LIM_2_TR
0	ERR_TR

Figure 3-18. Edit Terminal Values dialog, third version

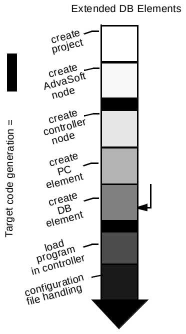
48. Fill in following information:

    Set the MMI_USE to YES.

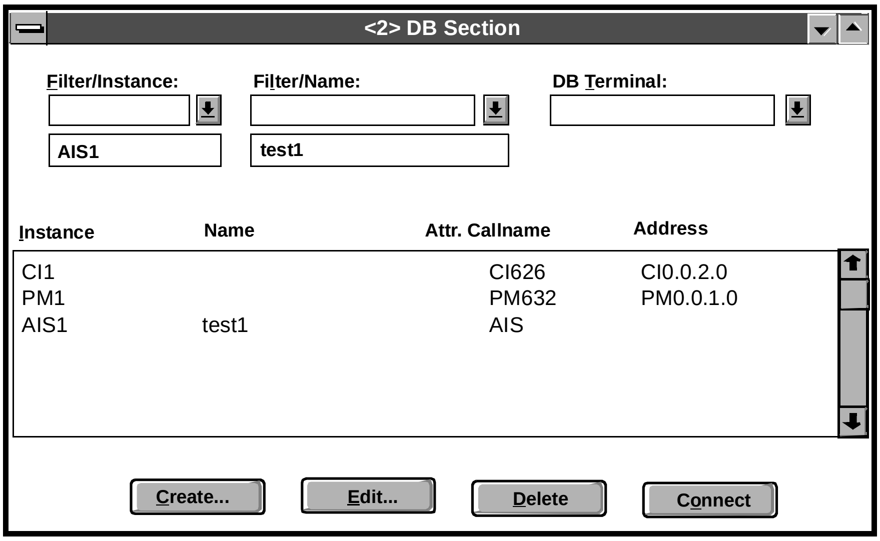
49. Click the OK button.

    You return to the Create Element dialog.

50. Click the Apply button in the Create Element dialog.
51. Choose the Save item from the File menu to save the configuration.
52. Choose the Generate Target Code item from the File menu.



During the last four steps, the DB Section dialog is updated continuously according to your selections.



Filter/Instance: AIS1

Filter/Name: test1

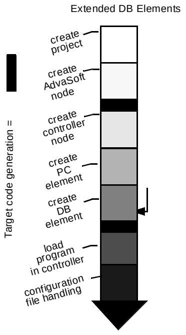
DB Terminal:

Instance	Name	Attr. Callname	Address
CI1		CI626	CI0.0.2.0
PM1		PM632	PM0.0.1.0
AIS1	test1	AIS	

Figure 3-19. DB Section dialog

53. Click the Close button in the Create Element dialog (if you click the OK button, an additional AIS is created).

    The Create Element dialog is closed and the DB Section dialog becomes the active dialog. The chart showing the different DB sections is updated with the items created during the steps above.

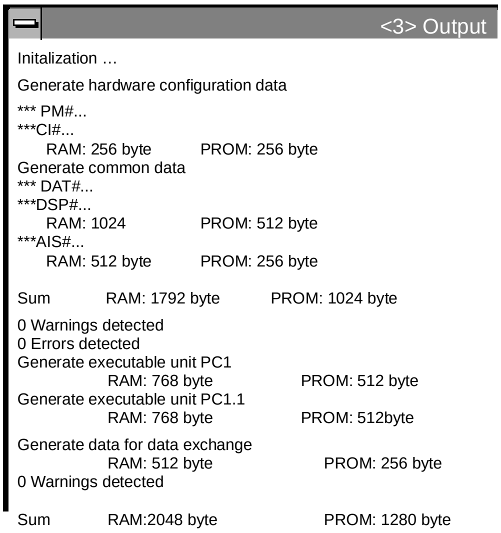


Compiling the controller application and BOIL file creation

54. You are still in the Function Chart Builder application. Choose Generate Target Code from the File menu.

    A dialog is displayed which asks you to confirm the changes made and to confirm that they are to be saved to the controller node you are creating. Click the YES button.

Your application is compiled. Brief information about the compiling procedure is displayed.

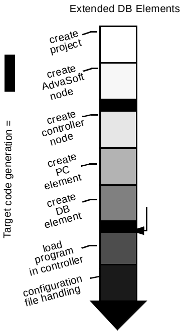


```
Initalization …
Generate hardware configuration data
*** PM#...
***CI#...
     RAM: 256 byte          PROM: 256 byte
Generate common data
*** DAT#...
***DSP#...
     RAM: 1024              PROM: 512 byte
***AIS#...
     RAM: 512 byte          PROM: 256 byte

Sum          RAM: 1792 byte          PROM: 1024 byte
0 Warnings detected
0 Errors detected
Generate executable unit PC1
                RAM: 768 byte             PROM: 512 byte
Generate executable unit PC1.1
                RAM: 768 byte             PROM: 512byte
Generate data for data exchange
                RAM: 512 byte                 PROM: 256 byte
0 Warnings detected

Sum             RAM:2048 byte                 PROM: 1280 byte
```

Figure 3-20. Example of the information displayed after a completed DB target code generation

NOTE

During the compilation, the BOIL file is created.



Downloading to controller

55. Choose the Block Program item from the Target menu in the Function Chart Builder application.

    Some information message boxes are displayed asking if you want to block program execution in the controller. Affirm all questions.

56. Choose Load Program from the Target menu.

    A dialog is displayed with various connection alternatives. In the case of a totally new AF100 Interface installation and a totally new controller, use a serial communication line. Check the appropriate COMx radio button (depending on which port you are using) and click OK. You can use the AF100 Interface in all future application program download operations.

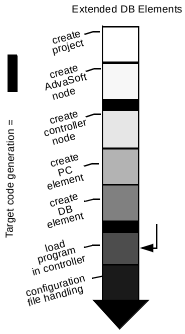
    The Function Chart Builder displays some messages about the program loading procedure’s progress.

57. Choose the Deblock item from the Tools menu.

58. Exit the Function Chart Builder application.

    A dialog is shown which asks if you want to disconnect the program currently executing in the controller. Click the YES button.

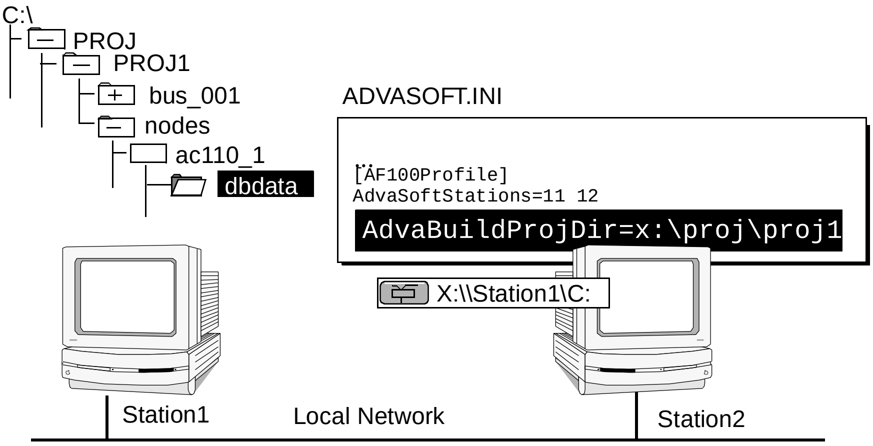
59. A dialog is displayed which states that the program is not saved in the PROM. Saving in the PROM is not necessary for this tutorial, so click the YES button.



Handling the configuration files

60. Create an identical directory tree in the AdvaSoft station (compared to the AdvaBuild station) and copy the created BOIL files to the dbdata directories.

    If the AdvaBuild application resides on the AdvaSoft station used for this tutorial, and the project directory is …\DBDATA, then it is not necessary to copy the BOIL file. The same applies when you are using a local network, for example, Microsoft Windows Network, with the path to the project directory specified in the ADVASOFT.INI file.



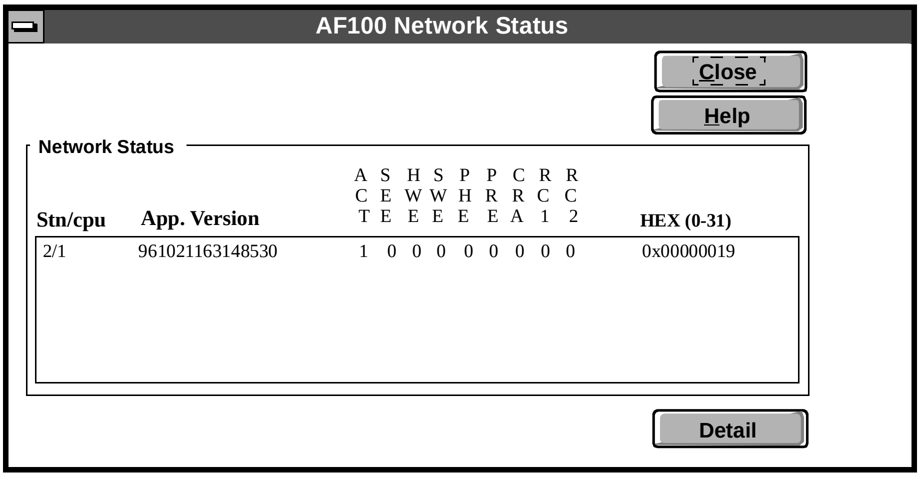
```
[AF100Profile]
AdvaSoftStations=11 12
AdvaBuildProjDir=x:\proj\proj1
```

Figure 3-21. How to transfer, or share, the configuration files between two workstations

Testing the communication 1

61. Start AdvaSoft for Windows.
62. Click the AF100 Interface icon (the program must be active) and select the Status item.
63. Click the Network button in the displayed dialog.

    A new dialog is displayed, showing the network nodes connected to the AF100 Interface bus and the node status.



Stn/cpu	App. Version	ACT	SEE	HWE	SWE	PHE	PRE	CRA	RC1	RC2	HEX (0-31)
2/1	961021163148530	1	0	0	0	0	0	0	0	0	0x00000019

Figure 3-22. AF100 Network Status dialog after a completed tutorial

For an explanation of the different dialog texts, see Section 4.4.2, Hardware and network diagnostics.

Testing the communication 2

You can try to access an object using a USER API application, for example, AdvaCommand or AdvaInform DDE Server. When you use the AdvaCommand application, do the following:

64. Start the AdvaCommand application by double-clicking the AdvaCommand icon in the AdvaCommand program group.
65. Open the DRAW application.
66. Create a link.

    Click the ABC/100 icon in the tool box. Click the stamp icon in the work space and fill in the following in the displayed Data Link dialog:

    STATION1:TEST1.F_CV (or the node entered in the SCU)

    and click OK.
67. Select the Quickview item in the View menu (Ctrl+Q).

    If the connection is functioning, the value 0.00 is shown.

3.4.2 Creating a small test configuration using DAT Based Objects

This tutorial is also useful to test that the AF100 Interface is up and running after the installation procedure is completed. To carry out this tutorial, the following applications and hardware must be installed and working:

- AdvaSoft for Windows, Version 1.3
- AF100 Interface, Version 1.2
- AdvaBuild for Windows, Version 1.3
- AdvaCommand for Windows, Version 1.2
- One AC110, Version 2.0, or later, with all options installed.

This tutorial assumes that this is a new installation with no previous node or network configurations. During the tutorial, a "procedure bar" is placed at strategic positions to indicate where in the procedure the tutorial is at any given moment.

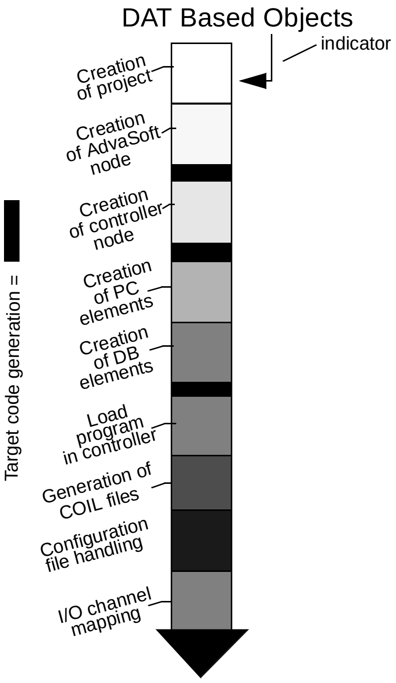
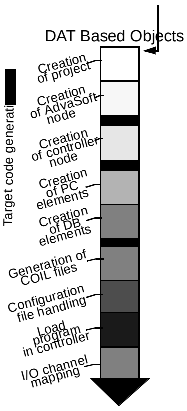


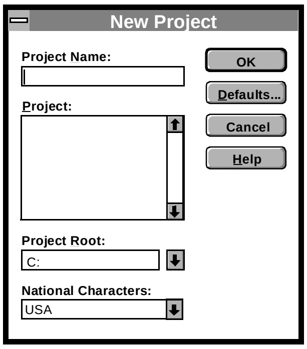
Figure 3-23. Procedure bar used for DAT Based Objects



Creating a new project

1. Start the application builder.
2. Select New Section/New Project from the File menu.

   The following dialog is displayed:



Project Name:

Project:

Project Root: C:

National Characters: USA

Figure 3-24. New Project dialog

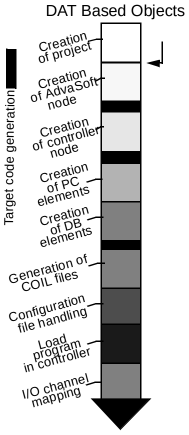
3. Fill in the following information in the New Project dialog:

   Set Project Name to AdvaSoft (if you use a project name other than AdvaSoft, you must manually edit the ADVASOFT.INI file).

   Set Project Root to C: (or as appropriate).
4. Click OK.

   Three windows are displayed:

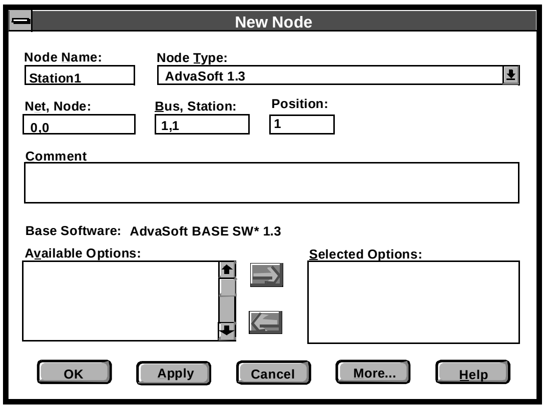
   - The Node List window
   - The Circuit List window
   - The Type Circuit List window.



Creating a new AdvaSoft station node

5. Select New/New Node from the File menu.

   The following dialog is displayed:



New Node

Node Name: Station1

Node Type: AdvaSoft 1.3

Net, Node: 0,0

Bus, Station: 1,1

Position: 1

Comment

Base Software: AdvaSoft BASE SW* 1.3

Available Options:

Selected Options:

Figure 3-25. New Node dialog

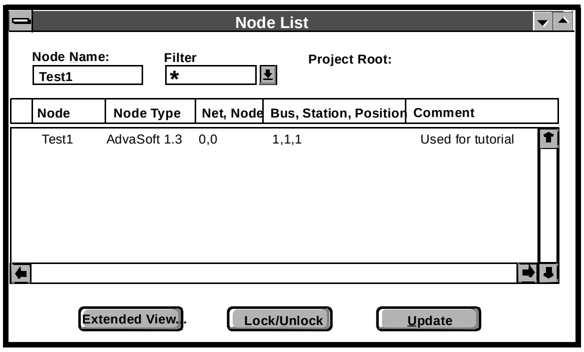
6. Fill in the following information in the New Node dialog:
   - Set Node Name as appropriate (Station1).
   - Set Node type to AdvaSoft 1.3.
   - Set Net, Node to 0,0.
   - Set Bus, Station as appropriate (1,1).

NOTE

It is important to remember the setting for the bus number when you are configuring the controller node. The station number must be unique for each station.

7. Click OK.

The Node List dialog is displayed:



Node Name: Test1

Filter: *

Project Root:

Node	Node Type	Net, Node	Bus, Station, Position	Comment
Test1	AdvaSoft 1.3	0,0	1,1,1	Used for tutorial

Figure 3-26. Node List dialog

8. Select the new AdvaSoft station node and choose the Function Chart Builder item from the Tools menu.

   The Function Chart Builder application is started. During the start-up sequence you are asked for the page size used for the chart diagram. When the start-up sequence is complete, a dialog with the program elements is displayed:

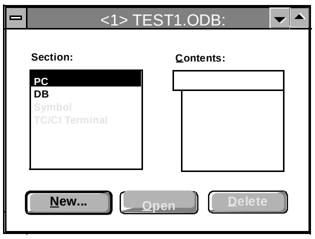


Figure 3-27. The <1> TEST1.ODB: dialog

9. Choose New Section /DB item from the File menu.

The Create Element dialog is displayed:

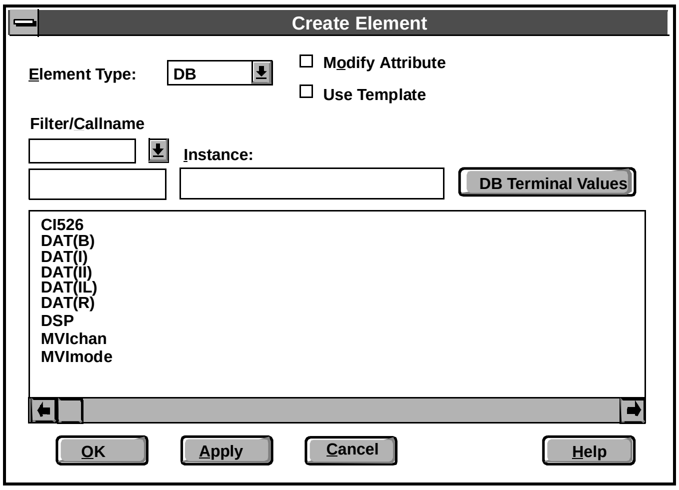
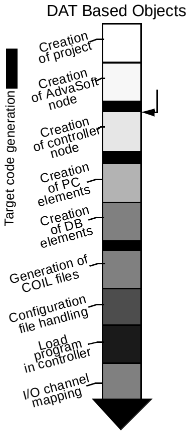


Figure 3-28. Create Element dialog, DB element version

10. In the list box, mark the CI526 (or higher) item and click the DB Terminal Values button.
11. Enter the station and bus number (1,1) and press OK.
12. Click the OK button and then click the OK button in the Create Element dialog.
13. Save the new node by choosing the Save item from the File menu.
14. Choose the Generate target code item from the File menu.

    A window is displayed with information about the outcome of the code generating procedure.

15. Exit the Function Chart Builder application.

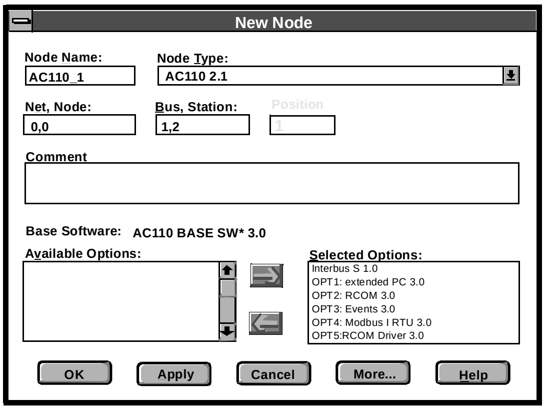


Creating a new controller node

16. Activate the AdvaBuild Application Builder.

17. Select the New/New Node item from the File menu.

    The following dialog is displayed (all fields are empty as default):



Node Name: AC110_1

Node Type: AC110 2.1

Net, Node: 0,0

Bus, Station: 1,2

Position: 1

Comment:

Base Software: AC110 BASE SW* 3.0

Available Options:

Selected Options:
Interbus S 1.0
OPT1: extended PC 3.0
OPT2: RCOM 3.0
OPT3: Events 3.0
OPT4: Modbus I RTU 3.0
OPT5:RCOM Driver 3.0

Figure 3-29. The "controller version" of the New Node dialog

18. Fill in the following information in the New Node dialog:
    - Set Node Name as appropriate (AC110_1).
    - Set Node type to AC110 2.1.
    - Set Net, Node to 0,0.
    - Set Bus, Station as appropriate (1,2).

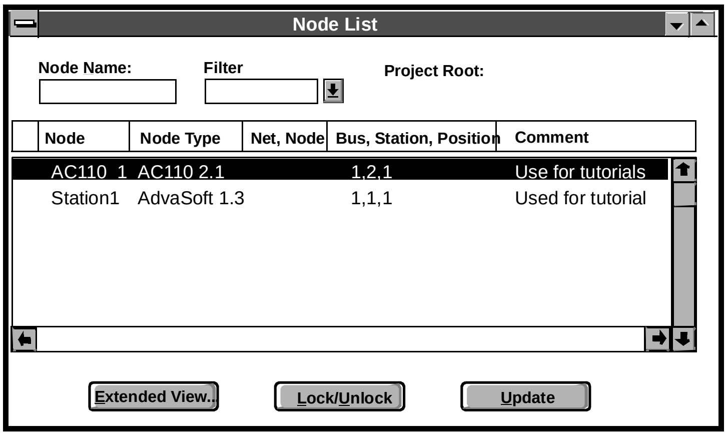
NOTE

The setting for the bus number must be the same as the setting for the AdvaSoft station with which the controller is to communicate (see earlier step in the tutorial). The station number must correspond with the station number setting in the controller.

19. Select all available options.

20. Click OK.

The node list dialog is displayed with the newly created controller node listed among the other items in the list box.



Node	Node Type	Net, Node	Bus, Station, Position	Comment
AC110_1	AC110 2.1		1,2,1	Use for tutorials
Station1	AdvaSoft 1.3		1,1,1	Used for tutorial

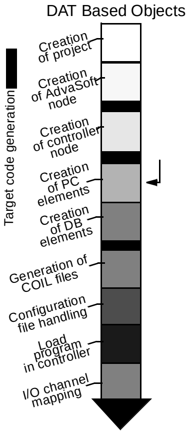
Figure 3-30. Node List dialog

21. Select the new controller node and choose the Function Chart Builder item from the Tools menu.

    The Function Chart Builder application is started.

22. Choose the Generate Target Code item from the File menu.

    New target code, including the newly created controller node, is created.

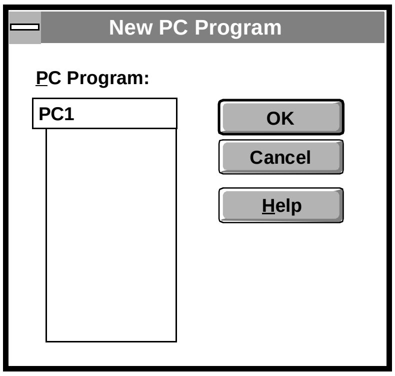


Building the controller application

23. Activate the Function Chart Builder application.

24. Choose New/PC from the File menu in the Function Chart Builder application.

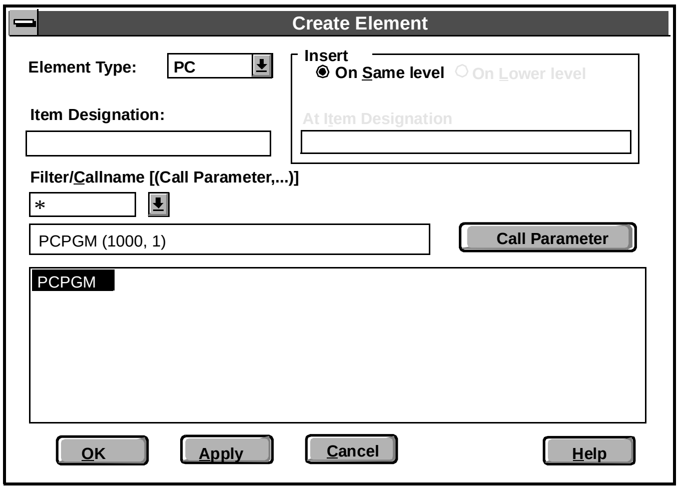
    The New PC Program dialog is displayed:



PC Program: PC1

Figure 3-31. New PC Program dialog

25. Fill in the name of your new PC program (PC1) and click OK.
26. The Create Element dialog is displayed.



Create Element

Element Type: PC

Insert: ● On Same level ○ On Lower level

Filter/Callname [(Call Parameter,...)]: *

PCPGM (1000, 1)

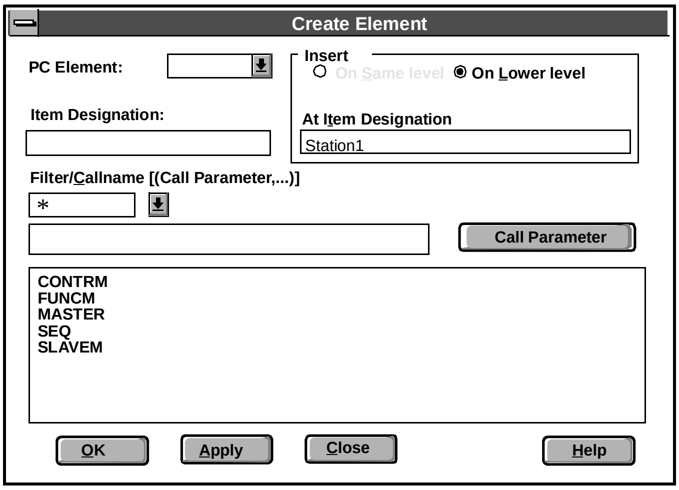
PCPGM

Figure 3-32. Create Element dialog for a PC element

27. Click on the PCPGM item in the list box.
28. Click the Call Parameter button.
29. Fill in the cyclicity time, for example, 100 ms, in the displayed dialog and click OK.

    You return to the Create Elements dialog.
30. Click the Apply button in the Create Elements dialog.

A new set of parameters is displayed in the list box:



At Item Designation: Station1

Filter/Callname: *

CONTRM
FUNCM
MASTER
SEQ
SLAVEM

Figure 3-33. Create Element dialog for a PC element, second version

31. Click the CONTRM item in the list box.
32. Click the Call Parameter button.
33. Fill in the cyclicity time, for example, 100 ms, in the displayed dialog and click OK.

    You return to the Create Elements dialog. The chart showing the configured program elements is updated with the CONTRM item.
34. Click the OK button.

    The FCB application displays a chart showing the items created during the steps above:

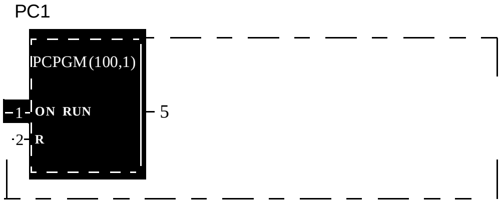


Figure 3-34. Example of the chart displayed after generating target code for a PC program

35. Activate the Chart window (PC Section - PC1) in the Function Chart Builder application, and click on the 1 connection, belonging to the PCPGM item, to mark it.
36. Choose the Connect item from the PC -Terminal menu.

37. In the To section of the displayed dialog, fill in the following:

    D=1
    and click the Apply/Next button.
    Fill in *D=0*
    click the Apply/Next button
    Fill in *D=1*
    click the Apply/Next button
    Fill in *D=0*
    click the Apply/Next button
    Fill in *D=0*
    click the OK button.

    After this procedure, the chart should look like this:

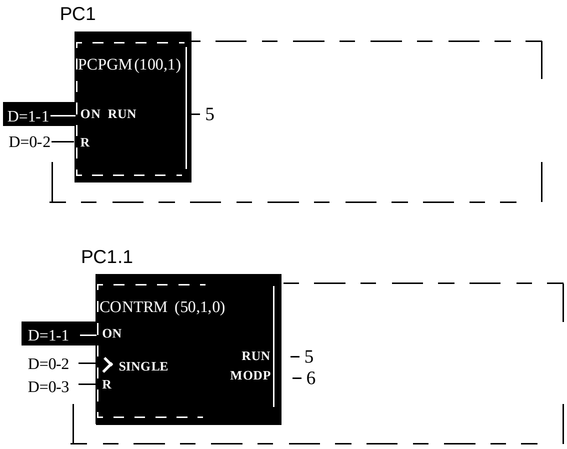
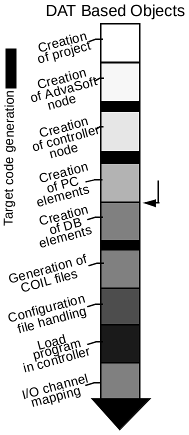
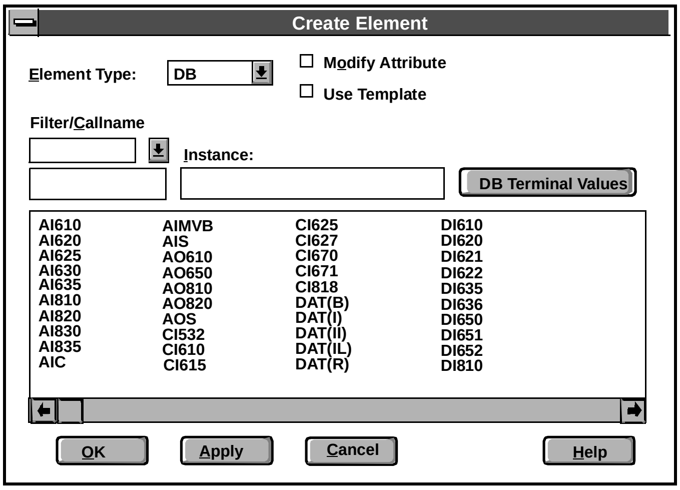


Figure 3-35. Chart after the connection procedure

38. Choose New/DB from the File menu.



The Create Element dialog is displayed:


Element Type: DB

☐ Modify Attribute

☐ Use Template

Figure 3-36. Create Element dialog, DB element version

39. In the list box, mark the CI626 (or appropriate) item and click the Apply button.
40. In the list box, mark the PM632 (or appropriate) item and click the Apply button.
41. In the list box, mark (click on) the DSP item and--this is important--click the DB Terminal Value button.

    The Edit DB Terminal Values - DSP1 dialog is displayed.

test1

Figure 3-37. Edit Terminal Values dialog

42. Fill in following information:

    Set the NAME item to an appropriate object name (TEST1).

    Set the IDENT item as appropriate (2).

    Set the NO_REAL to 1.

    Set the SOURCE item to SEND.

43. Click the OK button.

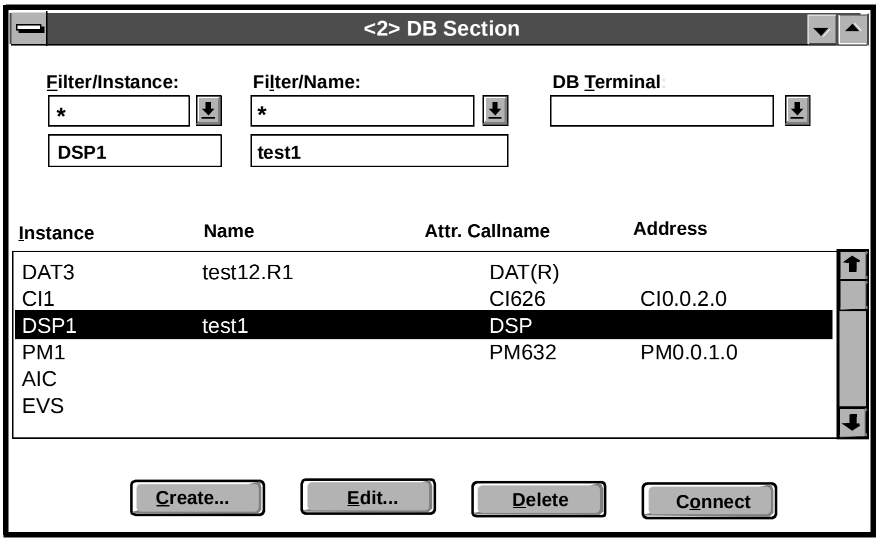
    You return to the Create Element dialog.

44. Click the Apply button in the Create Element dialog.

45. Click the Close button in the Create Element dialog (if you click the OK button an additional DSP is created).

    The Create Element dialog is closed and the DB Section dialog becomes the active dialog.

    The DB Section dialog is updated with your new DSP object and, it is important to note, an automatically created DAT object:



<2> DB Section

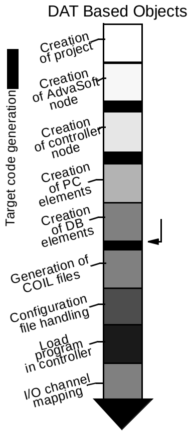
Filter/Instance: *
DSP1

Filter/Name: *
test1

DB Terminal:

Instance	Name	Attr. Callname	Address
DAT3	test12.R1	DAT(R)	
CI1		CI626	CI0.0.2.0
DSP1	test1	DSP	
PM1		PM632	PM0.1.0
AIC			
EVS			

Figure 3-38. DB Section dialog with the automatically created DAT element



46. Select the automatically created DAT element and click the Edit button.

    The Edit DB Terminal Values - <object name> is displayed.

47. Remove the period placed in the object name.

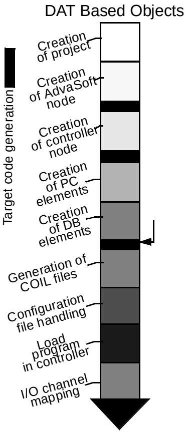
    As an example: change test12.R1 to test12R1.

NOTE

You remove the period because AdvaCommand for Windows uses the period as a separator between object and value.

48. Click the OK button.

    You return to the DB Section dialog.



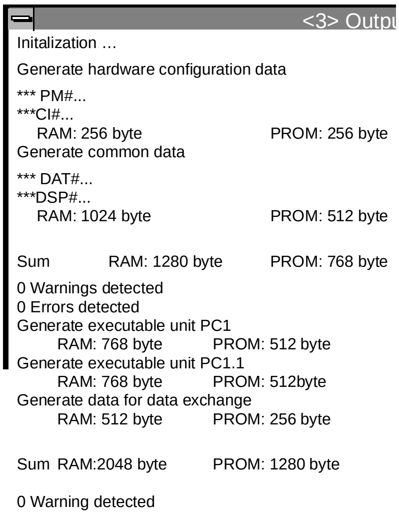
Compiling the controller application

49. Choose the Generate Target Code item from the File menu.

    A dialog is displayed asking if you want to save the detected changes in the PC and DB section(s).

50. Click the YES button in the dialog.

    The target code is generated, including the new DAT based objects item. Brief information about the compiling procedure is displayed.



```
Initalization ...

Generate hardware configuration data

*** PM#...
***CI#...
    RAM: 256 byte                  PROM: 256 byte
Generate common data

*** DAT#...
***DSP#...
    RAM: 1024 byte                 PROM: 512 byte


Sum          RAM: 1280 byte        PROM: 768 byte

0 Warnings detected
0 Errors detected
Generate executable unit PC1
        RAM: 768 byte        PROM: 512 byte
Generate executable unit PC1.1
        RAM: 768 byte        PROM: 512byte
Generate data for data exchange
        RAM: 512 byte        PROM: 256 byte


Sum  RAM:2048 byte           PROM: 1280 byte

0 Warning detected
```

Figure 3-39. Example of the information displayed after a code generation

51. Exit the Function Chart Builder.

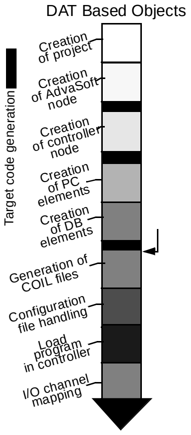
Downloading the controller application

52. Choose the Block Program item from the Target menu in the Function Chart Builder application.

    Some information message boxes are displayed which ask if you want to block program execution in the controller. Affirm all questions.

53. Choose Load Program from the Target menu.

    A dialog is displayed with various connection alternatives. In the case of a totally new AF100 Interface installation and a totally new controller, use a serial communication line. Check the appropriate COMx radio button (depending on which port you are using) and click OK. You can use the AF100 Interface in all future application program download operations.



The Application Builder displays some messages about the progress of the program loading procedure.

54. Exit the Function Chart Builder application.

    A dialog is shown which asks if you want to disconnect the program. Click the YES button.

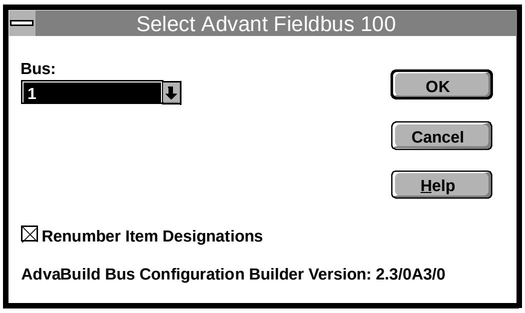
    A dialog is displayed stating that the program is not saved in the PROM. Saving in the PROM is not necessary for this tutorial, so click the YES button.

Generating COIL files

55. Activate the Application Builder.

56. Choose the Bus Configuration Builder item from the Tools menu.

    The Bus Configuration Builder application is started and the Select Advant Fieldbus 100 dialog is displayed.



Bus: 1

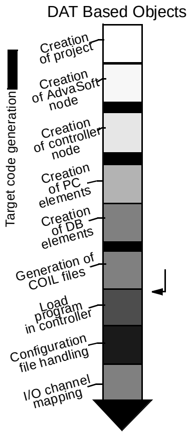
☒ Renumber Item Designations

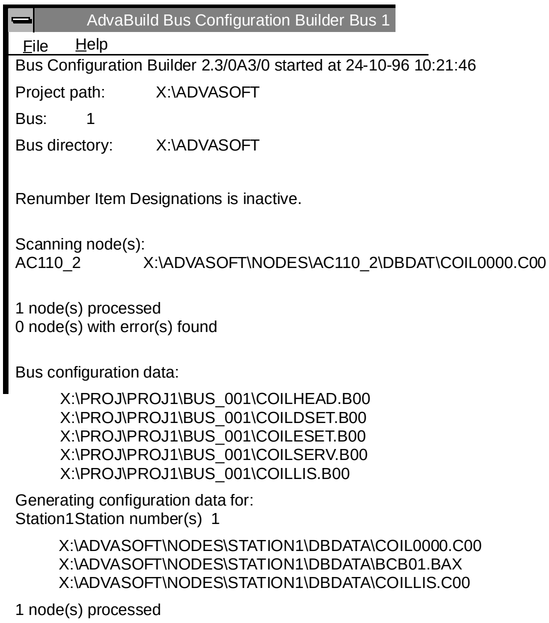
AdvaBuild Bus Configuration Builder Version: 2.3/0A3/0

Figure 3-40. Initial dialog in the Bus Configuration Builder application

57. Click the OK button.

The compilation procedure starts. Brief information about the completed procedure is displayed.

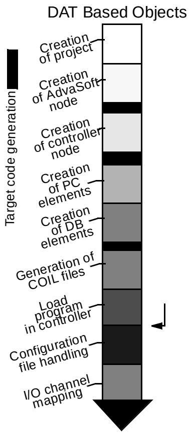




```
Bus Configuration Builder 2.3/0A3/0 started at 24-10-96 10:21:46
Project path:        X:\ADVASOFT
Bus:        1
Bus directory:       X:\ADVASOFT

Renumber Item Designations is inactive.

Scanning node(s):
AC110_2              X:\ADVASOFT\NODES\AC110_2\DBDAT\COIL0000.C00

1 node(s) processed
0 node(s) with error(s) found

Bus configuration data:
        X:\PROJ\PROJ1\BUS_001\COILHEAD.B00
        X:\PROJ\PROJ1\BUS_001\COILDSET.B00
        X:\PROJ\PROJ1\BUS_001\COILESET.B00
        X:\PROJ\PROJ1\BUS_001\COILSERV.B00
        X:\PROJ\PROJ1\BUS_001\COILLIS.B00
Generating configuration data for:
Station1Station number(s)  1
        X:\ADVASOFT\NODES\STATION1\DBDATA\COIL0000.C00
        X:\ADVASOFT\NODES\STATION1\DBDATA\BCB01.BAX
        X:\ADVASOFT\NODES\STATION1\DBDATA\COILLIS.C00
1 node(s) processed
```

Figure 3-41. Example of information displayed when building the configuration files

The information displayed also contains the directories where the different configuration files are placed.



58. Choose the Exit item from the File menu.

Handling of the configuration files

59. Copy the created COIL files to their default directories in the AdvaSoft station.

    If the "AdvaBuild Station" and the "AdvaSoft Station" are one and the same, then this operation is not necessary.

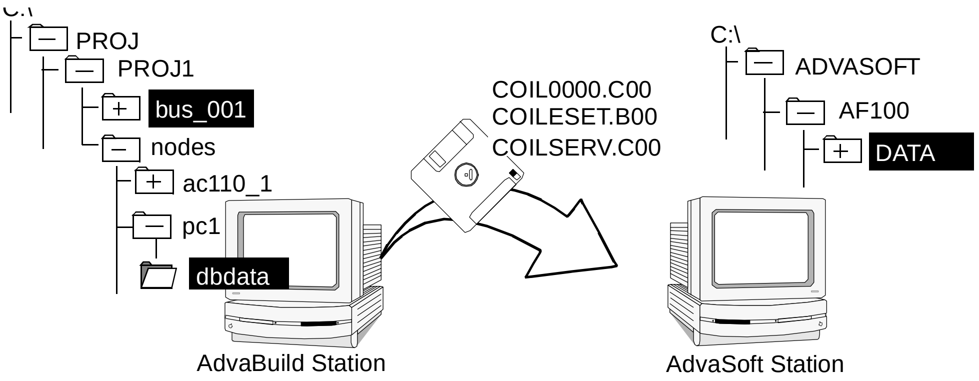
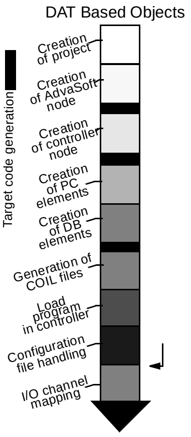


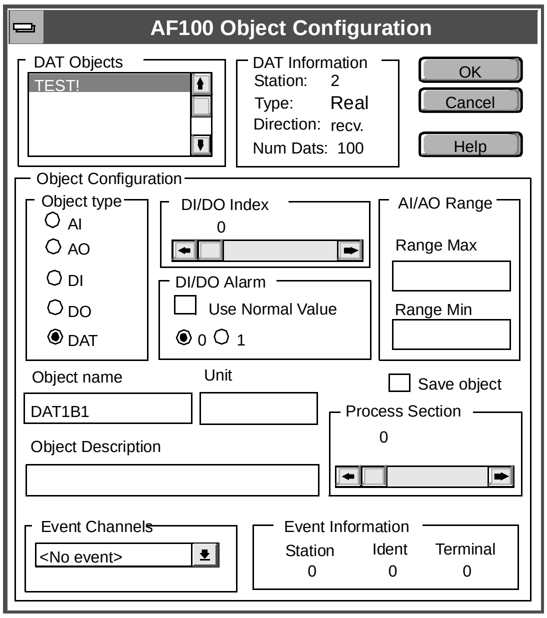
Figure 3-42. Example on how to distribute the configuration files



I/O channel mapping

60. In the program manager, activate the AF100 Object Config. 1.2/0 application in the AdvaSoft program group.

    The following dialog is displayed:



DAT Objects: TEST!

DAT Information
Station: 2
Type: Real
Direction: recv.
Num Dats: 100

Object Configuration

Object type: ○ AI ○ AO ○ DI ○ DO ● DAT

DI/DO Index: 0

DI/DO Alarm: ☐ Use Normal Value ● 0 ○ 1

AI/AO Range
Range Max:
Range Min:

☐ Save object

Object name: DAT1B1

Unit:

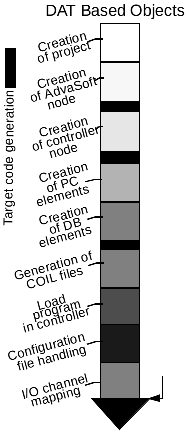
Object Description:

Process Section: 0

Event Channels: <No event>

Event Information
Station: 0
Ident: 0
Terminal: 0

Figure 3-43. AF100 Object Configuration dialog



The following steps describe the configuration of an AI object:

61. Select the TEST1 object in the **DAT objects** list box.

    Verify its source from the information in the DAT information section.
    Set the **Object type** to AI in the Object Configuration section.
    Enter the AI object name in the **Object name** text box or accept the suggested name (the suggested name for AI objects is the same as the DAT object name).

62. Click the OK button.

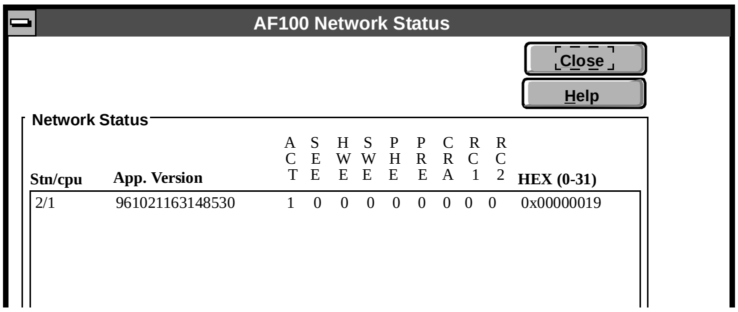
Testing the communication 1

63. Activate AdvaSoft for Windows.

64. Click the AF 100 Interface icon (the program must be active) and select the Status item.

65. Click the Network button in the displayed dialog.

    A new dialog is displayed that shows the network nodes connected to the AF100 Interface bus and the node status.



Stn/cpu	App. Version	ACT	SEE	HWE	SWE	PHE	PRE	CRA	RC1	RC2	HEX (0-31)
2/1	961021163148530	1	0	0	0	0	0	0	0	0	0x00000019

Figure 3-44. AF100 Network Status dialog after a completed tutorial

Testing the communication 2

You can try to access an object by using a USER API application, for example AdvaCommand or AdvaInform DDE Server. When you use the AdvaCommand application, do the following:

66. Start the AdvaCommand application by double-clicking the AdvaCommand icon in the AdvaCommand program group.

67. Open the Draw application.

68. Create a link.

    Click the ABC/100 icon in the tool box. Click the stamp icon in the work space and fill in the following in the displayed Data Link dialog:
    STATION1:TEST1.F_CV and click OK.

69. Select the Quickview item from the View menu (Ctrl+Q).

    If the connection is functioning, the value 0.00 is shown.

3.5 Application Procedures

3.5.1 Configuration and application design

Consider the following design aspects when creating configuration data and building applications:

- **Sending data**

  You can send commands to the process controller via the Service Data Protocol or via Data Set Peripherals.

  Normally, you use Service Data Protocol when sending commands from the AF100 Interface, see Figure 3-45.

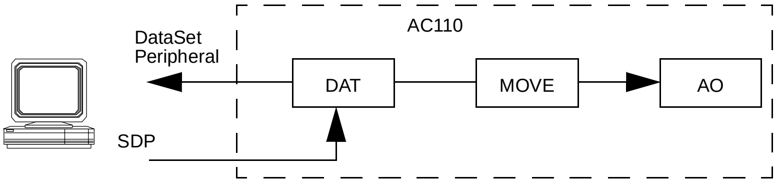


Figure 3-45. Commands sent via the Service Data Protocol

Service Data Protocol is used for all DAT objects defined in **receiving** DataSet Peripherals in the PC with the Advant Fieldbus 100 Interface.

You can update the DAT value from any PC with the Advant Fieldbus 100 Interface without any specific application in the process controller station.

NOTE

Only **one** DI/DO object can be mapped to a DAT object with this solution since the command sent via the Service Data Protocol affects all bits of the DAT.

You can also send data from the AdvaSoft station with DataSet Peripherals. This is the case for all DAT objects residing in the **sending** DataSet in a PC with the Advant Fieldbus 100 Interface.

These commands are sent cyclically, which must be handled by the remote controller station. Sending commands via DataSet Peripherals enables you to send the same command to multiple AC110 controllers.

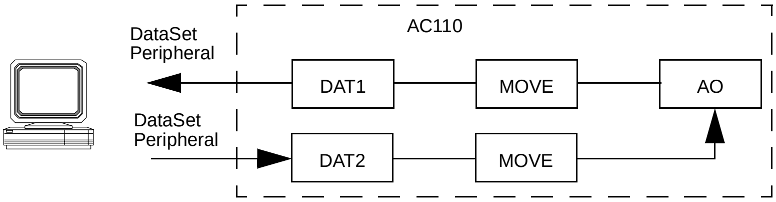


Figure 3-46. Commands sent via DataSet Peripherals

With this solution, you can use all bits of the DAT to map different DI/DO objects.

You can also use DataSet Peripherals to transfer data between PCs equipped with the Advant Fieldbus 100 Interface.

The differences between these two methods are summarized in the Table 3-5:

Table 3-5. Command methods

Command method	+	-
Service Data Protocol (SDP)	Multiple AdvaSoft stations can send commands to the same object without any additional logic in the controller.	No guarantee when the command is sent. Depends on the load in the PC.
DataSet Peripheral (DSP)	Commands are sent directly, independent of the load in the PC. A command can be "broadcasted" to multiple controllers (all configured with a receiving DSP for this command).	Needs additional logic in the AC110 to handle multiple AdvaSoft stations' commands to the same object. Requires additional DSPs (one for sending the value and one for receiving the mirrored value from the AC110).

- **Cyclic subscriptions**

  Subscription frequency is set in the client application. Subscription frequency that is higher than the frequency at which DAT values are updated on the CI board has no effect. Avoid this to minimize the AF100 Communication handler's load.

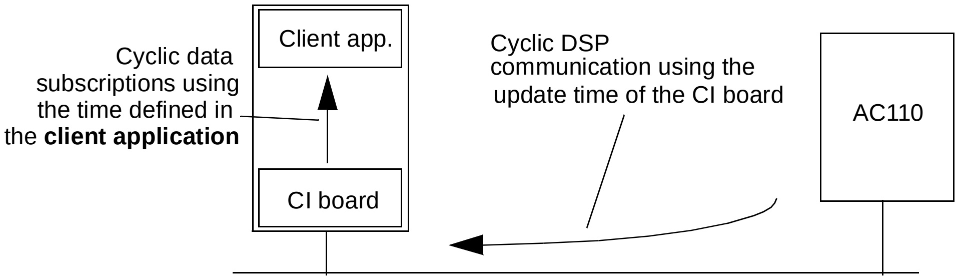

Cyclic data subscriptions using the time defined in the **client application**
Client app.
CI board
Cyclic DSP communication using the update time of the CI board
AC110

Figure 3-47. Two sources for cyclic subscription times

Refer to Section *4.2 Operating Overview* for more information on the runtime operation.

3.5.2 Extended DB Elements

3.5.2.1 Configuration data files using Extended DB Elements

If the AdvaBuild for Windows application is the only user of the Advant Fieldbus 100 Interface, then the only place you must define the station number is in the ADVASOFT.INI file.
If, however, other AdvaSoft for Windows clients, such as AdvaCommand, are used when accessing process data, you need a configuration data file. See below:

- **BOIL0000.C00**

  This file contains data for **Extended DB Elements** for the AF100 Interface. There is a BOIL file for every controller that supports Extended DB Elements and one BOIL file for each AdvaSoft for Windows station.

The AF100 Interface has a BOIL file handling function. This function scans the BOIL files and updates (or builds when no symbol file is present) the symbol file after application version changes. The symbol file contains all information about the EVSs, DSPs and Extended DB Elements in each BOIL file. The symbol file is in binary format for faster loading and access.

3.5.2.2 Generating configuration data files

When you are creating a configuration data file, the first step is to generate target code.
The target code contains information about the other stations on the Advant Fieldbus 100 network with which the Advant Fieldbus 100 Interface is to communicate.

To create the configuration data files, use the AdvaBuild for Windows application.

1. Run the **Application Builder** and create, if necessary, a new project and then a new node. The data for the new node is as follows:

   – Set **Name** to an appropriate name.
   It is recommended that you create a strategy for naming your network nodes.
   For example:

   AC110_2
   Station1

   etc.

   – Set **Node Type** to **AdvaSoft 1.3, 2.1 AC110** or as appropriate.

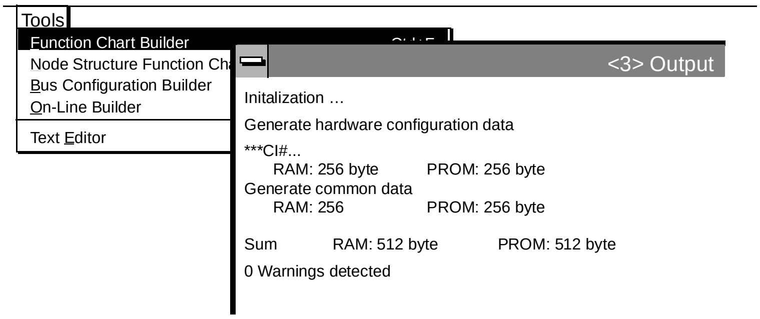
   – Set **Net, Node** to **0,0**.

   – Set **Bus, Station to <AF100 bus number>, <station number >**.

   The bus number is set to the same number as for the other stations in the project with which the Advant Fieldbus 100 Interface is to communicate. The number has nothing to do with the physical bus. The bus number provides you an alternative means for sectioning your AF100 network.

   Note that the station number for a controller must correspond with the number in the controller cabinet.

2. Run the **Function Chart Builder (FCB)** and generate target code.



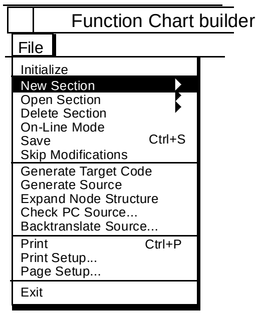
```
Initalization …
Generate hardware configuration data
***CI#...
     RAM: 256 byte          PROM: 256 byte
Generate common data
     RAM: 256               PROM: 256 byte

Sum          RAM: 512 byte           PROM: 512 byte
0 Warnings detected
```

Figure 3-48. Function Chart Builder can be started from the Tools menu in AdvaBuild

3. Repeat steps 1 and 2 for all nodes.
4. Build your controller application program in the **Function Chart Builder (FCB)**.

   Select the target controller from the node list and choose the Function Chart Builder item from the Tools menu. Then build your PC and DB elements for the controller. For a more thorough description of how to select the various PC elements and DB elements, see the tutorial in Section 3.4, Tutorial and Table 3-4.
5. Generate target code.
6. Repeat steps 4 and 5 for all controller nodes.

   During this process, the BOIL0000.C00 file is created and placed in the C:\PROJ\PROJ1\NODES\AC110_1\DBDATA directory.
7. Load the application to the controller.
8. Distribute the configuration files to the target stations.



3.5.3 DAT Based Objects

3.5.3.1 Configuration data files

DAT based objects offer a reduced set of parameters and functionality in comparison to Extended DB Elements. For this reason, use DAT based objects only when you are configuring a system already using this type of object or when the need for a maximum number of elements in a DataSet is the primary issue.

The AF100 Interface uses a number of configuration files when handling DAT based objects. These configuration files (COIL files) are used for CI board configuration, EVS configuration and object reference information. All of the COIL files are in ASCII format.

NOTE

> If AdvaBuild for Windows is the only user of the Advant Fieldbus 100 Interface, then the only place you must define the station number is in the ADVASOFT.INI file. If, however, other AdvaSoft for Windows clients, such as AdvaCommand, are used when accessing process data, then you need the COIL files.

The configuration data files are as follows:

- **COIL0000.C00**

  The COIL0000.C00 file contains configuration information for the CI board and for **DAT based objects**. It contains device-specific parameters, such as the station number, and information on all DataSet Peripherals and corresponding DAT objects that should be configured on the CI board.

- **COILSERV.B00**

  The COILSERV.B00 file contains reference information for DAT objects needed when sending commands using the Service Data Protocol (SDP).

- **COILESET.B00**

  The COILESET.B00 file contains information on all configured Event Sets (EVS) and the corresponding Event channels. This file may be empty if no EVS is used.

The configuration data files are in ASCII format and are created by AdvaBuild for Windows. The files are to be transferred (copied) to the ADVASOFT directory on the station equipped with the Advant Fieldbus 100 Interface.

3.5.3.2 Generating configuration data files

The first step in creating a configuration data file is to generate target code. The target code contains information about the other stations on the Advant Fieldbus 100 network with which the Advant Fieldbus 100 Interface is to communicate.

To create the configuration data files, use the AdvaBuild for Windows application.

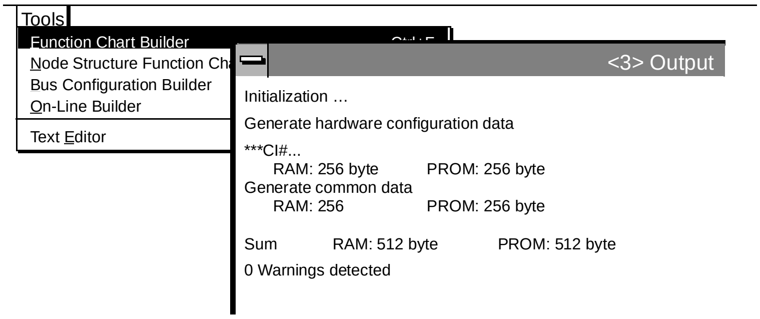
1. Run the **Application Builder** and create, if necessary, a new project and then a new node. The data for the new node is as follows:
   - Set **Name** to an appropriate name.
     It is recommended that you create a strategy for naming your network nodes.
     For example:

     AC110_2
     Station1

     etc.
   - Set **Node Type** to **AdvaSoft 1.3, 2.1 AC110** or as appropriate.
   - Set **Net, Node** to **0,0**.
   - Set **Bus, Station to <AF100 bus number>, <station number>**.

     The bus number is set to the same number as for the other stations in the project with which the Advant Fieldbus 100 Interface is to communicate. The number has nothing to do with the physical bus. The bus number provides you an alternative means of sectioning your AF100 network.

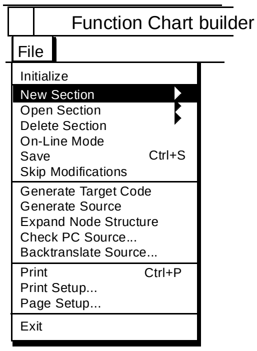
     Note that the station number for a controller must correspond with the number in the controller cabinet.
2. Run the **Function Chart Builder (FCB)** and generate target code.



```
Initialization …
Generate hardware configuration data
***CI#...
     RAM: 256 byte          PROM: 256 byte
Generate common data
     RAM: 256               PROM: 256 byte

Sum          RAM: 512 byte           PROM: 512 byte
0 Warnings detected
```

Figure 3-49. Function Chart Builder can be started from the Tools menu in AdvaBuild

3. Repeat steps 1 and 2 for all nodes.



4. Build your controller application program in the **Function Chart Builder (FCB)**.

   Select the target controller from the node list and choose the Function Chart Builder item from the Tools menu. Then build your PC and DB elements for the controller. For a more thorough description for how to select the various PC elements and DB elements, see the tutorial in Section 3.4, Tutorial and Table 3-4.

5. Generate target code.

6. Load the application to the controller.

   This is done in the FCB application.

7. Repeat steps 4-6 for all controller nodes.

8. Run the **Bus Configuration Builder (BCB)** program.

   This program reads the DataSet Peripheral and EventSet configuration for all stations defined on the same bus number. It creates configuration data files for the Advant Fieldbus 100 Interface. These files are placed in the following directories:

   – The COIL0000.C00 file resides in the **...\DBDATA** directory for the AdvaSoft target in the project.

   – The COILESET.B00 and the COISERV.B00 files reside in the **BUS_xxx** directory for the project, where xxx is the bus number defined in step 1.

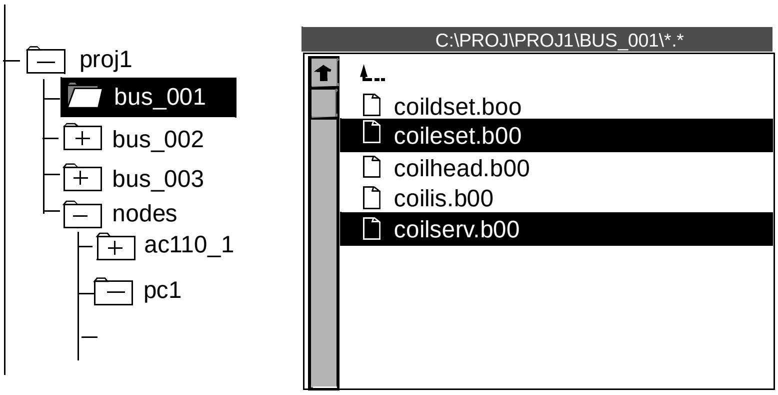


Figure 3-50. Example of the directory tree used for COIL files

   If you need to modify the configuration data files, for example, to delete DataSet Peripherals which are not of interest to the Advant Fieldbus 100 Interface, enter the FCB program. Backtranslate the source code for the applicable stations and edit the DSP database elements. Then generate target code again, which creates a new COIL0000.C00 file in the DBDATA directory.

9. Distribute the configuration files to the target stations.

Copy the configuration data files to the system directory for AdvaSoft for Windows, defined in the ADVASOFT.INI file (default directory is ADVASOFT\AF100\DATA).

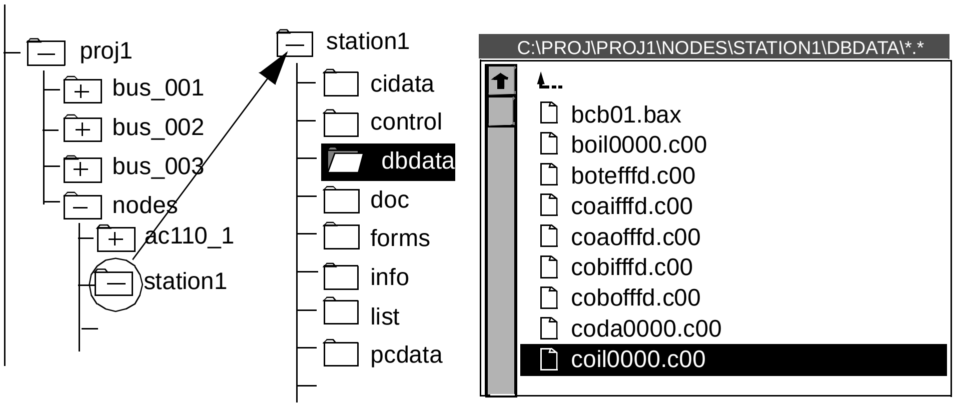


Figure 3-51. Directory tree required to make the AF100 Interface work

10. Make I/O channel mapping for all objects using the AF100 Object Configuration utility in the AdvaSoft target station.

    For more information about I/O channel mapping strategy and techniques for transferring object data over the AF100 Interface bus, see Section 3.1, Design Considerations and Section 3.5.4, Configuration of I/O channel for DAT Based Objects.

3.5.3.3 Speeding up the configuration process for multiple station systems

If you use more than one AdvaSoft station on the same AF 100 network, you can copy the configuration data from one station to the other stations. This reduces time-consuming work with the Object Configuration tool. Make your configuration as follows:

a. Make sure that you have the correct COILxxxx files present in all stations.

b. Do all configuration work with the Object Configuration tool in one AdvaSoft station.

c. Copy the file AF100SYM.DTA file (resides in directory ADVASOFT/AF100/DATA by default) from the configured station to your other AdvaSoft stations.

d. Resolve all objects in the AF100SYM.DTA file towards the COILxxxx files in the stations. To Resolve, select the Resolve item from the Control menu for the Advant Fieldbus 100 Communication program.

e. If you opt to send DataSet Peripherals (DSP) in the AdvaSoft stations (i.e., DSP corresponding to receiving DSP in the AC 110 nodes), configure all DAT-elements for the sending DSP with the Object Configuration tool.

The AdvaSoft station will have a high CPU load if non-configured Events are received. Non-configured Events are reported in the AdvaSoft Error Log. Always configure all Events with the Object Configuration tool or remove unused Event Sets from the COILESET.B00 file (resides in directory ADVASOFT by default).

3.5.3.4 Changing bus number

When you change bus numbers, delete the files located in c:\advasoft\af100\data directory by default:

af100evs.*
af100dsp.*
af100bo.*

This generates a new parsing of the BOIL files (similar to the Resolve operation done on DAT based objects).

3.5.4 Configuration of I/O channel for DAT Based Objects

The BCB application in AdvaBuild produces configuration files that contain information about all configured DAT objects in the DataSet Peripherals and Event channels in the Event Sets. The configuration files have no information on how the DataSet Peripherals and the Event Sets are related to each other. To configure a DAT based object, use the configuration tool to link the DSP, containing the DAT, with the EVS.

For example, see Figure 3-52 below.

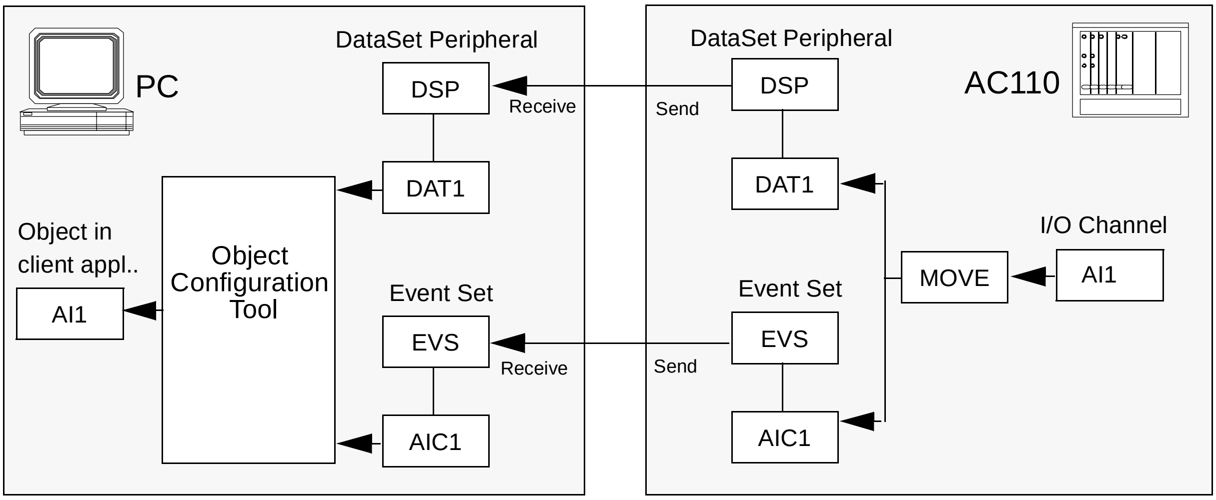


Figure 3-52. DataSet Peripheral and Event handling with the AF100 Interface

The value for the I/O channel AI1 is transferred via DAT1 in a DataSet Peripheral to the Advant Fieldbus 100 Interface. Time tagged events for AI1 are sent via an Event Set. Configure this event connection information using the Object Configuration utility included in the Advant Fieldbus 100 Interface.

The following functions are provided:

- Setup of I/O object type.

Object values are treated as DAT objects by default. You can map DATs to the correct I/O channel type: Analog Output (AO), Digital Input (DI) or Digital Output (DO).

The function to retrieve data from the controller is performed using DSPs. The DSP function uses DATs to transport the dynamic values. To have the functionality of AI, AO, DI and DO objects, map the DATs with this tool to the corresponding object type.

- Configure object attributes.

  Functions for configuring object attribute information (object description, engineering unit, process section, etc.) which cannot be received from the AF100 network are provided. Objects are, by default, accessed with the DAT object name. You can set a user-defined object name from this software.

- Connection to event channels.

  You must connect an object to the correct event channel to properly handle time tagged events generated for the object.

You can also use the Object Configuration utility to configure object attributes for DAT based objects not provided by the AdvaBuild-produced configuration data files.

You can configure the following attributes:

NOTE

Use the AF100 Object Configuration Utility for **DAT based objects** only. You cannot access or change **Extended DB Elements** from the AF100 Object Configuration utility.

- **Object name**

  Objects are, by default, accessed with the DAT object name. You can set up a user-defined object name, for example, the I/O channel name, from this software. The object name can be up to 20 characters.

- **Object type**

  Object values are, by default, treated as DAT objects. With the AF100 Object Configuration software, you can map the DAT objects to the correct I/O channel type (AI, AO, DI or DO).

- **Object description**

  You can create a description text for the object of up to 28 characters with this software.

- **Engineering unit**

  The engineering unit can be up to six characters. If you use AdvaCommand, limit the length to four characters.

- **Process section**

  You can define process sections from 0 to 16. Process section 0 disables process sectioning.

- **Value ranges**

  You can define the minimum and maximum value ranges for the object.

- **Normal value**

  If you desire alarm generation, define the normal value for DI/DO objects. An alarm is generated when the value is changed from the normal value, and an event is generated when the value returns to normal. If a normal value is not used, events are generated for value changes in both directions.

Note that not all object attributes are supported by all object types. Engineering unit and value ranges are only supported by AI/AO objects. The normal value is only supported by DI/DO objects. For DAT objects, only the object name can be defined.

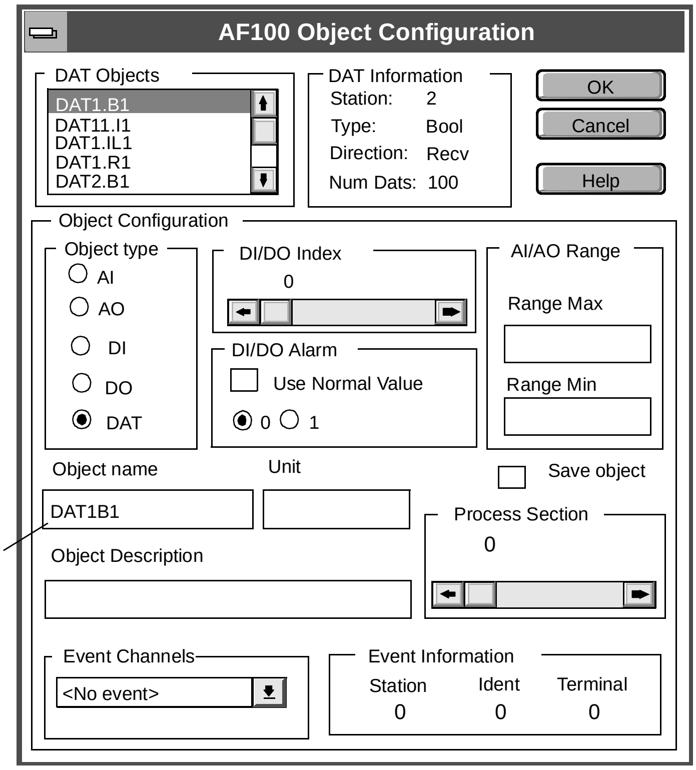
3.5.4.1 Using the AF100 Object Configuration software

Use the AF100 Object Configuration utility to define object configuration data not provided by the FCB-produced COIL files. Observe that the AF100 Object Configuration software is only for configuration of DAT based objects.

To carry out the configuration, the AF100 Object Configuration program reads the files containing configuration data for the DAT objects:

- AF100SYM.DTA
- COIL0000.C00 (creation described in the previous sections)
- COILSERV.B00 (creation described in the previous sections)
- COILSET.B00 (creation described in the previous sections)
- ADVASOFT.INI

The information retrieved from the configuration files is compiled and displayed in a dialog box, see Figure 3-53.



AF100 Object Configuration

DAT Objects
DAT1.B1
DAT11.I1
DAT1.IL1
DAT1.R1
DAT2.B1

DAT Information
Station: 2
Type: Bool
Direction: Recv
Num Dats: 100

Object Configuration

Object type
○ AI
○ AO
○ DI
○ DO
● DAT

DI/DO Index
0

DI/DO Alarm
☐ Use Normal Value
● 0 ○ 1

AI/AO Range
Range Max
Range Min

☐ Save object

Object name: DAT1B1

Unit

Object Description

Process Section
0

Event Channels
<No event>

Event Information
Station 0
Ident 0
Terminal 0

Note that you have to remove the period (.) in the object name

Figure 3-53. AF100 Object Configuration program

User-defined information provided during the configuration session is stored in a new configuration file named AF100SYM.DTA. This file is in a binary format. Whenever you use the Advant Fieldbus 100 Configuration program, a backup of the previous version of the AF100SYM.DTA file is created and named AF100SYM.BAK.

You can use the AF100 Configuration program in runtime together with the AF100 Communication program, which is automatically updated with the new user-defined information.

The program checks that an object name is not used twice. It also checks that correct object type mapping is performed, for example, that a DI/DO object is mapped to BOOL type DATs and AI/AO objects are mapped to REAL type DATs.

NOTE

If an object's object type is modified during runtime, the client application must restart all subscriptions for that object.

Configuration of AI/AO objects

These steps describe the configuration of an object to AI/AO object:

1. Select an object in the **DAT objects** list box.
2. Verify its source from the information in the DAT information section.
3. Set the **Object type** to AI/AO in the Object Configuration section.
4. Enter the AI/AO object name in the **Object name** text box or accept the suggested name (the suggested name for AI/AO objects is the same as the DAT object name).
5. Enter an object description in the **Object Description** text box.
6. Enter the engineering unit in the **Unit** text box.
7. Specify a process section with the **Process Section** scroll bar.
8. Enter the minimum and maximum ranges of the object value in the **Range Min** and **Range Max** text boxes.
9. Assign the object to an event channel by selecting one from the **Event Channel** combo box. This is optional.
10. Verify the information in the **Event Information** section.

Configuration of DI/DO objects

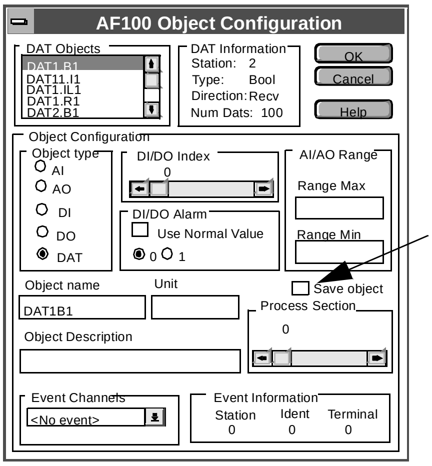
These steps describe the configuration of an object to DI/DO object:

1. Select an object in the **DAT objects** list box.
2. Verify its source from the information in the DAT information section.
3. Set the **Object type** to DI/DO in the Object Configuration section.
4. Select a Boolean signal from the current DAT object with the **DI/DO index** scroll bar.
5. Enter the DI/DO object name in the **Object name** text box or accept the suggested name. The suggested name for DI/DO objects is the same as the DAT object name with the extension "_<bit no.>" equalling its bit position selected with the DI/DO index scroll bar.
6. Enter an object description in the **Object Description** text box.
7. Specify a process section with the **Process Section** scroll bar.
8. Assign the object to an event channel by selecting one from the **Event Channel** combo box.
9. Select the **Use Normal Value** check box if alarm handling is required for the currently selected Boolean signal. Set the normal value to 0 or 1, and the signal goes into alarm condition when changing from normal state. Leave the check box blank if event handling without alarming is required. This step is optional.
10. Verify the information in the **Event Information** section.
11. Check the **Save object** check box.

Configuration of DAT objects

These steps describe the configuration of an object to DAT object:

1. Select an object in the **DAT objects** list box.
2. Verify its source from the information in the DAT information section.
3. Enter the DAT object name in the **Object name** text box or accept the suggested name (the suggested name for DAT objects is the same as the DAT object name).
4. Check the **Save object** check box.



DAT Objects: DAT1.B1, DAT11.I1, DAT1.IL1, DAT1.R1, DAT2.B1

DAT Information
Station: 2
Type: Bool
Direction: Recv
Num Dats: 100

Object type: ○ AI ○ AO ○ DI ○ DO ● DAT

DI/DO Index: 0

DI/DO Alarm: ☐ Use Normal Value ● 0 ○ 1

Object name: DAT1B1

☐ Save object

Process Section: 0

Event Channels: <No event>

Event Information: Station 0, Ident 0, Terminal 0

Do not forget to check this box when configuring DAT objects

Figure 3-54. AF100 Object Configuration dialog

NOTE

Though the only thing you can modify for a DAT object is the object name, it is important to note the following: When you configure a DAT object with the configuration tool, **always click the "Save object"** box in the AF100 object configuration dialog before you configure the next object.

Exit configuration

Click the **OK** button to save and exit. Click the **Cancel** button to exit without saving changes.

3.6 Configuration/Application Building Menus

There is no functionality in the AF100 Interface that supports Configuration/Application Building Menus.

Chapter 4 Runtime Operation

4.1 Product Operation

This section briefly describes how the Advant Fieldbus 100 Interface operates during start-up, runtime and shutdown.

CI board Start-up

When you start the AF100 Interface software (the communication part):

- The CI board initiates and the CI-profile, containing station number, bus number, etc., is downloaded. Detailed explanations of possible CI-profile sources follow.

  There are three sources for the CI board profile:

  – the BOIL0000.C00 file of the local AdvaSoft station

  – the ADVASOFT.INI [AF100Profile] section

  – the COIL0000.C00 file.

The flow diagram below defines the algorithm by which the profile is chosen.

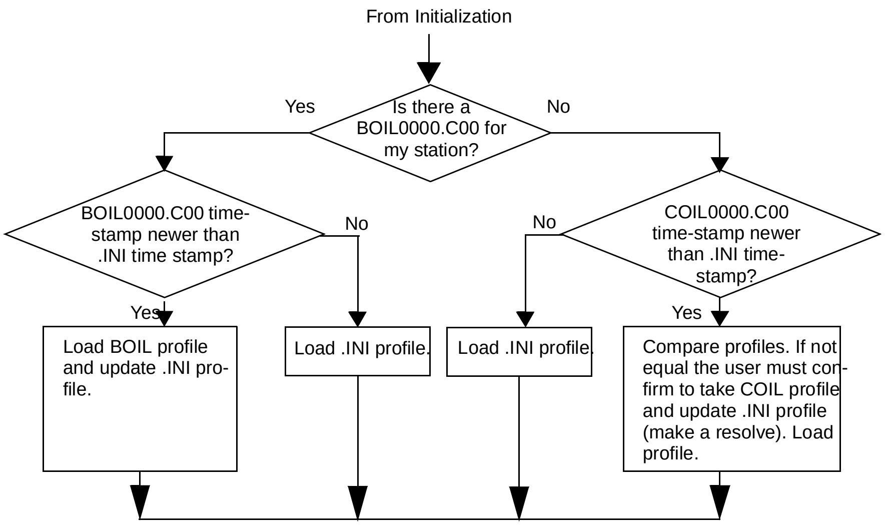

From Initialization
Yes
Is there a BOIL0000.C00 for my station?
No
BOIL0000.C00 time-stamp newer than .INI time stamp?
No
No
COIL0000.C00 time-stamp newer than .INI time-stamp?
Yes
Yes
Load BOIL profile and update .INI profile.
Load .INI profile.
Load .INI profile.
Compare profiles. If not equal the user must confirm to take COIL profile and update .INI profile (make a resolve). Load profile.

Figure 4-1. Flow diagram defining the method for selecting CI profile source

- All DSPs from the COIL and BOIL files are loaded (this step is omitted if the CI board is already operational when the software is started and/or if no new DSP configuration is to be loaded).

- All the EVSs from the COIL and BOIL files are loaded.
- The CI board goes operational and data exchange with DataSet Peripherals starts.
  The Application Version Control is activated.
- If the QuickLoad function is configured, all objects defined in the AF100SYM.DTA file are loaded into memory. Refer to Appendix A, Settings in the ADVASOFT.INI File.

Runtime

During runtime, the DATs on the CI board are cyclically updated via DataSet Peripheral (DSP) from the other stations on the Advant Fieldbus 100 network. The cycle time for the DSP communication is defined in configuration data files.

Cyclic subscriptions of DAT values are defined by the client application and handled by the AF100 Communication handler. They have no connection to the DSP cycle time.

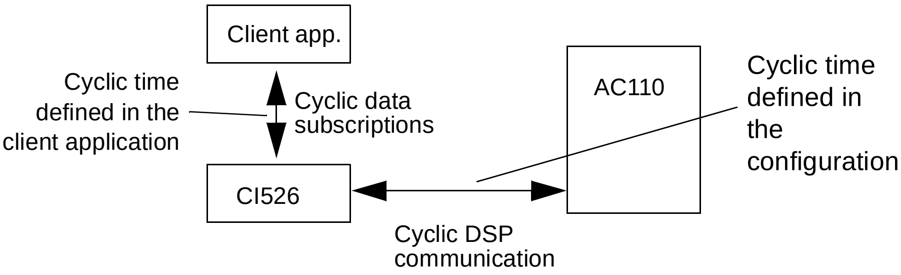

Client app.
Cyclic time defined in the client application
Cyclic data subscriptions
CI526
AC110
Cyclic time defined in the configuration
Cyclic DSP communication

Figure 4-2. Cyclic data traffic for the Advant Fieldbus 100 Interface

Shutdown

When you close the Advant Fieldbus 100 Interface software, the CI board goes to a passive state and the DataSet Peripheral transfer stops. The same happens if you press the **Disconnect** button in the **Configuration** dialog box.

The CI board also goes to a passive state if the AF100 Interface lifesign stops.

Security

Hardware specific configuration is stored both in the COIL0000.C00 file (generated from the FCB) and in the ADVASOFT.INI file (when user modified). If the parameters differ between these files, a message box appears which asks you which configuration to use. If the COIL0000.C00 file configuration is to be used, the ADVASOFT.INI file is updated.

There is no security, such as login with password, in the AF100 Interface when the object or hardware configuration is being modified.

4.2 Operating Overview

During runtime, the AF100 Communication software provides the following operator functions:

- Runtime configuration of the CI board.
- View of status and error messages from the CI board.
- Resolve of all objects defined in the AF100SYM.DTA file.
- View of current version of the Advant Fieldbus 100 Interface.

4.3 Runtime Tutorial

Not applicable.

4.4 Operating Instructions

Normally, you start or stop the AF100 Interface software by starting or stopping the client application using AdvaSoft for Windows. You can also control it directly by double-clicking the AF100 Interface icon from the Microsoft Windows desktop.

Shut down the software by shutting down the client application. You can also shut down the software by clicking the Close item in the Control menu. Note, however, that this does not affect the CI board operation. To stop CI board operation, you must power off the PC.

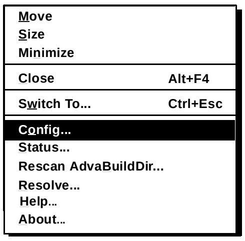
NOTE

If the startup takes a long time and/or there are bad object references in some displayed pictures, it is recommended that you delete the following files (located in the *…\advasoft\af100\data directory*):

af100evs.*
af100dsp.*
af100bo.*

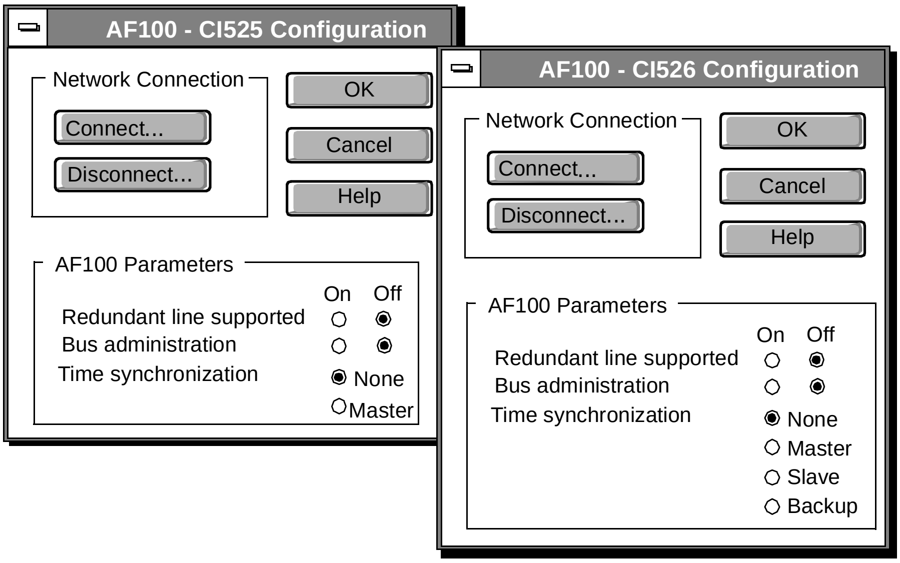
This generates a new parsing of the BOIL files (similar to the Resolve operation done on DAT based objects).

4.4.1 Runtime configuration of the CI525 and the CI526 interface boards



The AF100 interface software enables you to view and modify the configuration of the CI board in use. To accomplish this, the AF100 Interface uses a set of dialog boxes to display the information.

Click the **Config...** item from the AF100 Interface icon **System** menu. The AF100 Interface software displays one of the following **Configuration** dialog boxes, depending on the CI board in use, see Figure 4-3.


AF100 - CI525 Configuration

Network Connection
Connect...
Disconnect...

OK
Cancel
Help

AF100 Parameters

	On	Off
Redundant line supported	○	●
Bus administration	○	●

Time synchronization
● None
○ Master

AF100 - CI526 Configuration

Network Connection
Connect...
Disconnect...

OK
Cancel
Help

AF100 Parameters

	On	Off
Redundant line supported	○	●
Bus administration	○	●

Time synchronization
● None
○ Master
○ Slave
○ Backup

Figure 4-3. AF 100 configuration dialog boxes

The information displayed in the dialog box is not automatically updated if the station goes from one state to another. For a new update, reopen the dialog box.

The section for Network Connection features two buttons for connecting and disconnecting the CI board from the Advant Fieldbus 100 network.

- **Connect**

  Use the **Connect** button after a disconnect to reconnect the CI board to the Advant Fieldbus 100 network. The board is set to active state and goes operational again.

- **Disconnect**

  Use the **Disconnect** button to set the CI board in passive state. This disconnects it from the Advant Fieldbus 100 network. All configuration data on the CI board is preserved.

In the AF100 Parameters section, you can modify device-specific parameters for the CI interface board:

- **Redundant line supported**

  The redundant line support is set **On** if redundant TC625 modems are used.

- **Time synchronization**
  - CI525

    Set the time synchronization **On** if the Advant Fieldbus 100 Interface is to be clock master on the Advant Fieldbus 100 network. Otherwise, set the time synchronization to None. Note that only one station can be defined as time master in a network.

NOTE

When a CI525 board is used, the AF100 Interface cannot be configured as clock slave, i.e., time in the PCs equipped with AF100 Interface software will not be synchronized by the clock master on the AF100 network.

  - CI526

    Sets the Time Synchronization mode of the Advant Fieldbus 100 Interface. Note that only one station can be defined as master and only one station can be defined as backup.

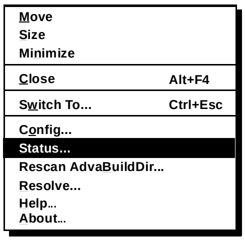
- **Bus administration**

  The bus administration is set **On** if the Advant Fieldbus 100 Interface is to be bus master on the Advant Fieldbus 100 network. This is also the normal configuration for the CI interface board. The Advant Fieldbus 100 network can have multiple bus masters. For more information, refer to the *Advant Fieldbus 100 User's Guide*.

Press the **OK** button to download new configuration parameters to the CI board and to the ADVASOFT.INI file. This information is used whenever you restart the AF100 Interface. Press the **Cancel** button to cancel the configuration.

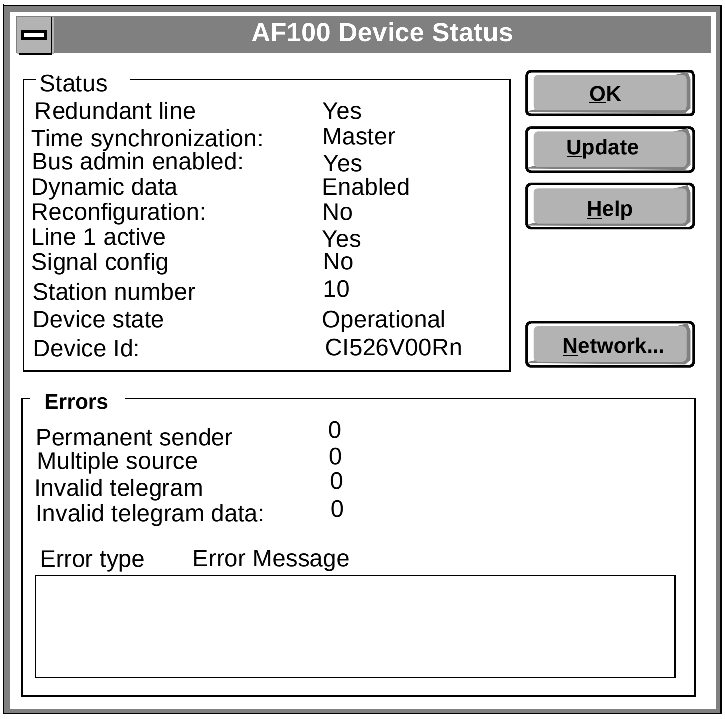
Press the **Help** button for on-line help information.

4.4.2 Hardware and network diagnostics



The AF100 Interface provides hardware and software performance supervision and error detection functions. Information is displayed via a number of dialog boxes which you can reach from the **System** menu.

Click the **Status...** item from the AF100 Interface icon's **System** menu to open the **AF100 Device Status** dialog box.



Status

Redundant line	Yes
Time synchronization:	Master
Bus admin enabled:	Yes
Dynamic data	Enabled
Reconfiguration:	No
Line 1 active	Yes
Signal config	No
Station number	10
Device state	Operational
Device Id:	CI526V00Rn

Errors

Permanent sender	0
Multiple source	0
Invalid telegram	0
Invalid telegram data:	0

Error type	Error Message

Figure 4-4. AF100 Device Status dialog box

The Status section gives the following information:

- **Redundant line**

  The Redundant line item shows whether a redundant line is configured.

- **Time synchronization**

  – **CI525 board**
  Possible modes are Master or None.

  - **CI526 board**
    Possible modes are None, Slave, Master or Backup. The current time synchronization mode does not necessarily equal the ClockMaster item in the ADVASOFT.INI file because of the time master backup functionality.

- **Bus admin enabled**

  The bus administration item shows if the CI is configured as a bus master on the Advant Fieldbus 100 network.

- **Dynamic data**

  The Dynamic data item shows whether DSP communication is enabled or disabled on the Advant Fieldbus 100 network. It should always be enabled.

- **Reconfiguration**

  The Reconfiguration item shows when errors or discrepancies are detected when the CI is scanning bus administration lists from other bus masters.

- **Line 1 active**

  The Line 1 active item shows that line 1 is chosen from the two redundant lines. When this item is **No**, line 2 is configured.

- **Signal config**

  The Signal config item shows that configuration data is being transferred to the CI board and normal runtime operation is disabled.

- **Station number**

  The Station number item shows the unique user-defined Advant Fieldbus 100 network station number.

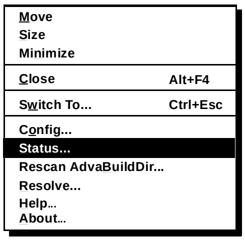
- **Device state**

  The Device state item shows whether the CI board is running properly. **Operational** means that the communication board is running. Other states (for example, **Error** or **Ready**) mean that the board is passive and not operating properly.

- **Device Id**

  The Device Id shows the type, version and revision of the interface board.

Error information



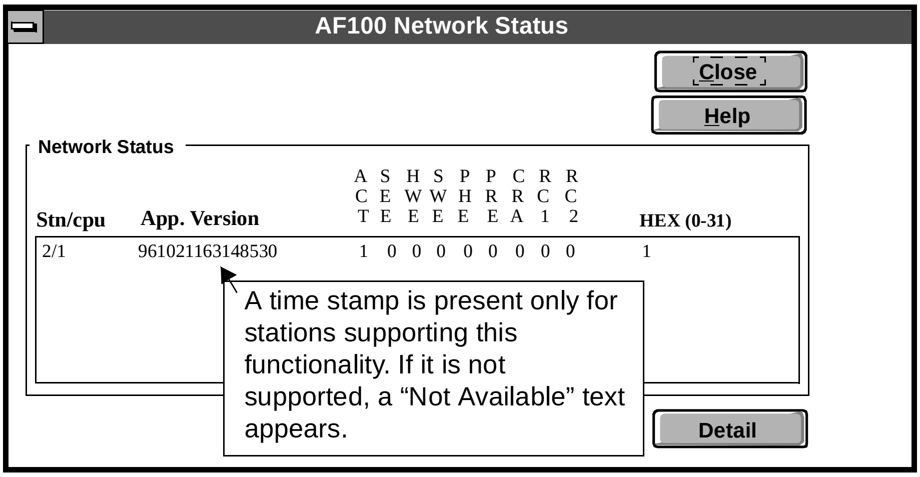
Errors are reported back to the common **System Messages** application included in the AdvaSoft for Windows Package. To view reported errors, open the **System Messages** application.

Errors from the CI board are presented in the listbox in the Errors section. Error information is explained in Section 5.3, Error Messages.

Network status

After you press the **Network** button, a dialog box with available stations on the Advant Fieldbus 100 network, and their Station Status, is presented. Note that this information is only valid for stations supporting the StationObject DSP (AC110s equipped with the CI625 bus coupler do not support this function).

The displayed network information, shown when you press the Network button, may look as follows:



Stn/cpu	App. Version	ACT	SEE	HWE	SWE	PHE	PRE	CRA	RC1	RC2	HEX (0-31)
2/1	961021163148530	1	0	0	0	0	0	0	0	0	1

A time stamp is present only for stations supporting this functionality. If it is not supported, a “Not Available” text appears.

Figure 4-5. Network Status dialog box, basic display

The information displayed in the listbox is updated cyclically.

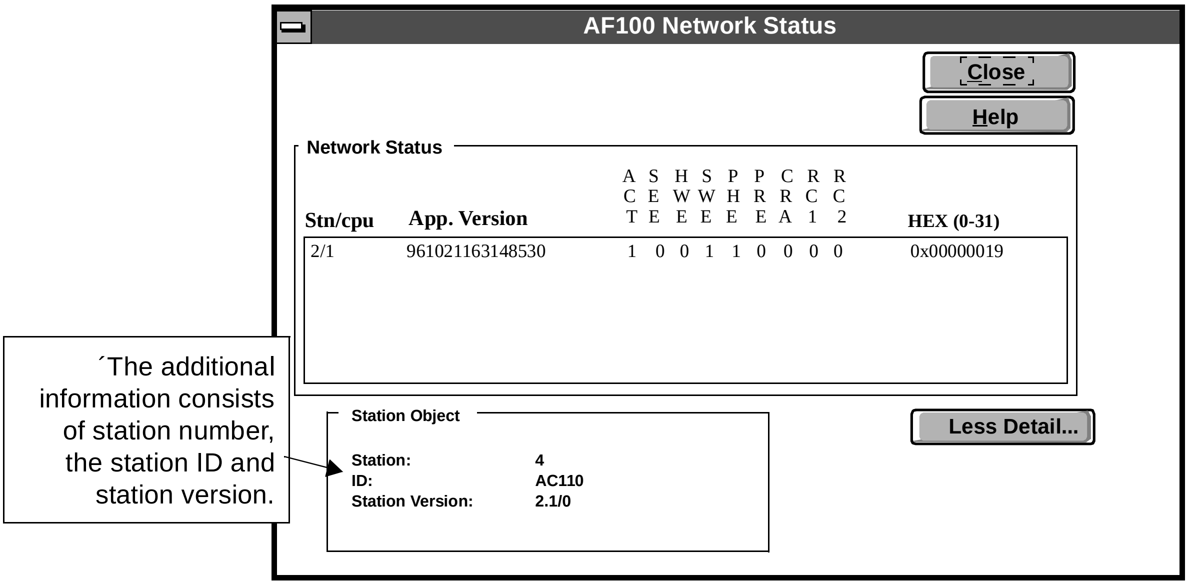
The Network Status section contains following information:

Stn/CPU	The node is displayed as Station number/CPU position.
App. (-lication) Version	Application version date and time
ACT	The station is active and executing the application in normal mode.
SE	Severe CPU error
HWE	Minor CPU hardware error
SWE	CPU software error
PHE	I/O device- or system error or missing I/O modules
PRE	I/O process error
CRA	CI device specific parameter: Set if redundant line supervision is active.
RC1, RC2	CI625 specific parameter: “redundant line failed” or “line 1 active” of CI625 status word.
HEX (0-31)	This is the complete status word including the bits described above.

All the above-mentioned information is updated cyclically.

Press the **Help** button to open context-sensitive help. Press the **Close** button to close the dialog and return the focus to the **Device Status** dialog box. Scrollbars are added automatically if the number of rows exceeds the space available in the list box.

Press the **More detail...** button to enhance the dialog box by displaying all additional information provided by the controller. Double-click a row in the list box to update the information in the additional **Station Object** box.



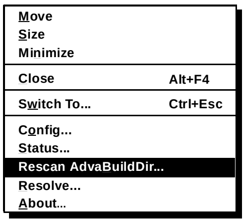
Stn/cpu	App. Version	A C T	S E E	H W E	S W E	P H E	P R E	C R A	R C 1	R C 2	HEX (0-31)
2/1	961021163148530	1	0	0	1	1	0	0	0	0	0x00000019

Station: 4
ID: AC110
Station Version: 2.1/0

´The additional information consists of station number, the station ID and station version.

Figure 4-6. Enhanced AF100 Network Status dialog box

4.4.3 Rescan AdvaBuildDir

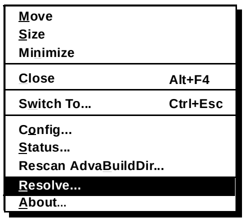


The AF 100 Interface has a feature for scanning the project directory tree for new nodes so you can add a new node without restarting.

1. Produce a new BOIL file, as described in Section 3.5, Application Procedures.
2. Copy the new configuration files, and the directory structure, from the AdvaBuild station to the target AdvaSoft station. This is done with the AF 100 Interface still running.
3. Choose the Rescan AdvaBuildDir… item from the AF100 Interface system menu.

   The new node is added and the information exchange is started.

4.4.4 Resolve



Select the **Resolve** item from the AF100 Interface icon **System** menu to resolve all DAT based objects in the AF100SYM.DTA file versus the FCB produced configuration files (COIL0000.C00, COILSERV.B00 and COILESET.B00). The resolve functions verify that the object exists in the COIL files and then retrieve the object references (DSP identity, DAT position, etc.) from these files.

Use the resolve function whenever new COIL files are generated (e.g., when new DSPs are added or modified in the AC110).

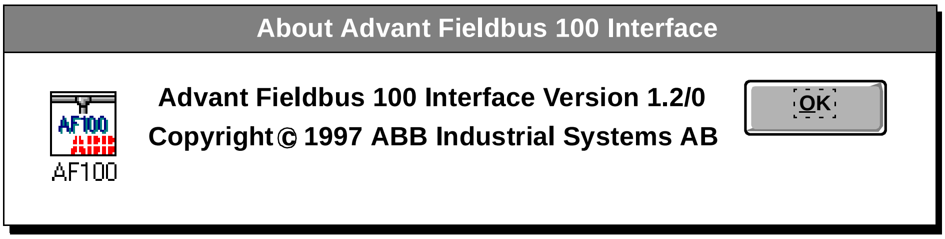
The resolve function first makes a backup of the current AF100SYM.DTA file to **AF100SYM.OLD**. Objects defined in the AF100SYM.DTA file that no longer exist in the COIL files are written in ASCII format to the **AF100SYM.ERR** file.

NOTE

The resolve function takes time, particularly when you use large configuration files. During resolve, dynamic data update for client applications is stopped.

4.4.5 Advant Fieldbus 100 Interface version

Select the **About...** item from the **Control** menu to view the current version of the Advant Fieldbus 100 Interface.



Advant Fieldbus 100 Interface Version 1.2/0
Copyright © 1997 ABB Industrial Systems AB

Figure 4-7. Advant Fieldbus 100 Interface product verification

4.5 Runtime Operation Menus

The runtime operation for the Advant Fieldbus 100 Interface is achieved from its **System** menu. Click once on the AF100 Interface icon.

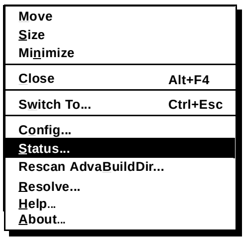


Figure 4-8. Advant Fieldbus 100 Interface System Menu

- **Move**
  is a standard Windows function for moving icons or application windows on the desk.
- **Close**
  terminates the Advant Fieldbus 100 Interface software.
- **Switch to...**
  is a standard Windows function for switching between MS-Windows applications.
- **Config...**
  opens the hardware configuration dialog box.
- **Status...**
  opens the hardware Status and Diagnostics dialog box.
- **Rescan AdvaBuildDir…**
  is used when new nodes are added during runtime.
- **Resolve**
  resolves the objects in the AF100SYM.DTA file towards the AdvaBuild produced configuration files.
- **Help...**
  starts the AF100 on-line help.
- **About...**
  opens a dialog box showing the Advant Fieldbus 100 Interface software's version and revision numbers.

Chapter 5 Maintenance

5.1 Preventive Maintenance

Follow ESD precautions to avoid component damage when you handle the CI board. For example, place the board on anti-static material and take grounding precautions.

We recommend that you follow a backup routine to secure configuration data regularly and whenever you reconfigure the system or change object definitions.

Make backups of the following configuration files:

- AF100SYM.DTA
- COIL0000.C00
- COILSERV.B00
- COILESET.B00
- BOIL0000.C00

For a general description of maintenance of the Advant Fieldbus 100 network, see the *Advant Fieldbus 100 User's Guide*.

5.2 Hardware Indicator

LED indicators on the modem indicate the following:

- The green **P** LED is the power indicator.
- The yellow **RX** LED is the bus receive indicator. It should be lighted when the modem is connected to the Advant Fieldbus 100 network cable.
- The yellow **TX** LED is the bus transmit indicator. It should start to blink when the CI board is up and running.

5.3 Error Messages

Error messages from the Advant Fieldbus 100 Interface are reported in two ways.

- **Software Errors:**
  All software errors are reported to the System Messages application, common to all AdvaSoft for Windows applications.
- **Hardware Errors:**
  Error information from the CI board is listed in the **Status** dialog box.

5.3.1 CI Errors

The Errors information from the CI board helps determine the source of errors.

- The **Permanent sender** identification shows the address of the permanent sender on the bus.
- The **Multiple source** item identifies the address of the station where two stations answer.
- The **Invalid telegram** item identifies the address of the station attempting to send non-identifiable information.
- The **Invalid telegram data** item shows the data content of the invalid telegram identified above.

The error messages listed below appear in the **Status** dialog box in the error message section. Please perform actions only if you are qualified to do so. If you send a problem report to your local ABB service organization, it is essential that you know the sequence of events before the error occurred and any error status and/or messages present. This information localizes the problem quickly and accurately.

Generally, errors are classified as follows:

- **Process errors,**
  usually caused by disturbances from the process or physical connections.
- **Fatal device errors,**
  usually caused by hardware failure.
- **System errors,**
  usually caused by faulty configuration or the combination of incompatible devices.

Process Error: More than 50% bus errors

Cause: More than 50% of the messages sent on the bus are faulty. Results in automatic reset of the CI board.

Action: Check physical bus components and connections.

Process Error: Simultaneous bus masters

Cause: Two or more bus masters are in the active state at the same time. Results in automatic hardware reset of the CI board.

Action: Check that two stations do not have the same station number configured.

Process Error: Redundant line failed

Cause: One of the communication lines on the bus has failed. The CI board continues operation.

Action: Check physical connections and modem operation of the failed bus line.

Device Error: Time latency error

Cause: Indicates that it was not possible to finish some of the bus administrator actions before the next timer interrupt. If this error is detected more than once in a 24-hour period, the CI board enters the 'ERROR' state. Assume hardware error on the CI board.

Action: Replace the CI board.

Device Error: Wrong DMA-channel active

Cause: Indicates that the wrong DMA-channel was active upon entry in the timer interrupt handler. The CI board enters the 'ERROR' state if this error occurs more than once in a 24-hour period. Assume hardware error on the CI board.

Action: Replace the CI board.

Device Error: Redundant line failed

Cause: One of the communication lines on the bus has failed. The CI board continues operation.

Action: Check physical connections and modem operation of the failed bus line.

Device Error: Fatal bus disturbance

Cause: Indicates no traffic has been detected on the bus, and the CI board is unable to hear itself when sending on the bus.

Action: Check physical bus components and connections.

Device Error: BAP failure

Cause: Indicates that the bus interface chip on the CI board is unable to determine its own bus address. The CI board enters the 'ERROR' state upon occurrence of this error. Assume hardware error.

Action: Replace the CI board.

Device Error: BAP-DMA error

Cause: Indicates that the bus interface chip could not access the DMA or data exchange memory fast enough. When this error occurs more than once in a 24-hour period, the CI board enters the 'ERROR' state. Assume hardware error.

Action: Replace the CI board.

Device Error: No time master

Cause: Indicates that no time is distributed on the bus. Assume that no CI boards have been assigned to be time master.

Action: Assign a CI board as time master.

System Error: Wrong parameter memory

Cause: Indicates that the table containing the information about the configured DSPs was invalid. The configuration is cleared in the CI board.

Action: Ignore message when occurring at cold start since there is no bus administrator list configured yet. If this message appears after DSP configuration is complete, reconfigure. If this is unsuccessful, exchange card.

System Error: No station configured

Cause: Indicates that the CI board station address is not configured.

Action: Configure the station address in the ADVASOFT.INI file.

System Error: Bus administration version conflict

Cause: Indicates that a protocol or generation version conflict has been detected on the CI board. This occurs, for example, when a CI625 and CI626 of incompatible versions, both with enabled bus administration functions, are connected to the same bus.

Action: Check for conflicting CI board versions, replace or set one in foreign mode, disabling bus administration functions.

System Error: Invalid config table

Cause: Indicates that the bus master configuration (DSP) table was invalid or that an invalid table was received from another bus master. Results in clearing of the table in the CI board.

Action: Reconfigure and restart the CI board.

System Error: Invalid master frame

Cause: Indicates that a device has attempted to send an invalid message at the end of the binary search device polling algorithm. This results in either sending of service data or DSP as events.

Action: Determine the address source and data content of the **Invalid telegram ID** and **Invalid telegram data** information boxes, if it has been possible to detect these. Reconfigure DSP list in this station if necessary.

System Error: Multiple devices

Cause: Indicates that two or more CI boards share the same station address.

Action: Check in the **Multiple source ID** information box and reconfigure one of the sources with another unique station address.

System Error: Permanent sender detected

Cause:	Indicates that the bus administrator has detected a permanent sender when it receives more than one answer to event polling. Assume hardware failure in one of the answering stations. One more try is attempted, and if the same station answers, it is classified as a permanent sender. It is identified in the **Permanent sender ID** information box. The bus administrator continues even though a permanent sender is identified.
Action:	Replace the CI board identified as the permanent sender.

System Error: Too many signals

Cause:	Indicates that too many signals have been configured. The bus administrator continues to operate without sending the last configured signals.
Action:	Reconfigure with fewer signals or increase cycle time of some signals.

System Error: Synchronization lost

Cause:	Indicates that the synchronized time could not be detected in the expected time slot. The bus master reverts to the init state. The error state persists until synchronization has been obtained.
Action:	Self-correcting.

System Error: Simultaneous time master

Cause:	Indicates that a collision is detected when performing time synchronization. This happens when more than one device is configured as clock master on the bus.
Action:	Reconfigure one or more devices so that only one clock master exists on the bus.

5.4 Fault-Finding and User Repair

If your system is running, but not properly, follow the steps listed below to determine the cause and the actions to take:

1. View the **System Messages** application from your Microsoft Windows desktop to determine the source of the problem.

   If the Advant Fieldbus 100 Interface software is the source, go on to the next step. If the problem originates from one of the other software units in your system, consult the corresponding user's guide.

2. Check the **CIxxx status** and check for critical status flags and error messages. Execute actions as instructed in Section 5.3, Error Messages.

   If the device state is not **operational**, try to restart the CI board by performing a Disconnect and then a Connect from the Config dialog box for the Advant Fieldbus 100 Interface.

3. If the above actions are not successful, contact your local ABB Industrial Systems support representative, as instructed in the *AdvaSoft for Windows User's Guide*.

If the Advant Fieldbus 100 Interface program is not responding, press **<Ctrl-Alt-Del>** and the PC restarts without resetting the CI board. If this has no effect, power off and power on your PC, simultaneously resetting the CI board.

Appendix A Settings in the ADVASOFT.INI File

A.1 General

The ADVASOFT.INI file, which resides in the MS-Windows directory, contains the [AF100Profile] section where you can set up a number of configuration parameters. As a default, this section contains an empty line for 'AdvaSoftStations' (described below), shown in Figure A-1.

Note that the parameters and values found in the .INI file are not case-sensitive.

```
...
[AF100Profile]
AdvaSoftStations=
```

Figure A-1. Default definitions in the ADVASOFT.INI file

You can define the following parameters in the [AF100Profile] section.

- **Station**
  This line is automatically added if you enter the station number in the installation dialog box during the AF 100 Interface software installation. You can also add this parameter manually by editing the ADVASOFT.INI file. Note that the station number must be in the range from 1 to 80.

  When you use COIL files generated by AdvaBuild, the station number in that file is copied to the ADVASOFT.INI file.

- **AdvaSoftStations**
  Defines other AdvaSoft stations on the Advant Fieldbus 100 network. Synchronizes acknowledgment of alarms among all PCs where the Advant Fieldbus 100 Interface is used. The AdvaSoft stations are separated with blank characters. You can leave this parameter empty if you use only new controllers.

- **Objects**
  Initial number of objects that can be handled. If more objects are loaded, more space for the data structure containing the objects is allocated automatically. If you omit this line, the default of 1000 objects is pre-allocated. If you subscribe to fewer objects, you can reduce this value to save allocated memory on your PC.

- **EventBufferSize**
  Size of the event buffer queue in this ASfW station. If this line is omitted, the event buffer queue provides space for 200 events.

- **EventBufferUpperLimit**
  Upper limit on size of the event buffer queue in this ASfW station. If the queue size transgresses this limit, a disable event command is sent to the controller. An enable event command is sent when the lower limit (see next point) is transgressed. If this line is omitted, the upper limit is 0.75 * `EventBufferSize`.
- **EventBufferLowerLimit**
  Lower limit on size of the event buffer queue in this ASfW station. If this line is omitted, the lower limit is 0.25 * `EventBufferSize`.
- **IRQlevel**
  Interrupt level of the CI board. Must be set up according to CI board strapping and can have the values 3, 10, 11, 12 or 15. If you use default CI board strapping (IRQ 11), you can omit this line.
- **IoBaseAddress**
  I/O address used by the C board. Must be set up according to CI board strapping. If you use default CI board strapping (I/O address 210 hex), you can omit this line.
- **QuickLoad**
  During startup of the AF100 software, the QuickLoad parameter preloads all objects defined in the AF100SYM.DTA file into memory and stores them there permanently. This may give faster response times for dynamic data update for the client applications. The QuickLoad function is, however, not activated by the AdvaBuild option since this option does not use the AF100 Interface object handling functions.
- **AdvaBuildProjDir**
  The path to the directory where AdvaBuild stores the BOIL files of this project. This is the top-level directory of a fixed directory structure

  `(<AdvaBuildProjDir>\nodes\<nodename>\dbdata\BOIL0000.C00)`

  defined by AdvaBuild.
- **Nodes**
  The names of the nodes (i.e., `<nodename>`s, see above) that should be considered by BOIL File Handling. If this entry is empty, then all nodes found in `<AdvaBuildProjDir>\nodes` are considered. Otherwise only the given nodes are considered.
- **UserEventTreat**
  Controls the generation and the eventTreatment parameter of the following events:
  - block/deblock process update
  - block/deblock alarm
  - AI object's alarm limit changed by operator

  If StatusEventTreat is set to "0", all of the above events are suppressed. If StatusEventTreat is set to something other than 0, the events are passed to the user and the eventTreatment is set to StatusEventTreat.

  "0" is used as default if this parameter is omitted.
- **ConfigTimeStamp**
  Configuration time stamp read from the COIL files.

- **BUS**
  BUS number used for this station.
- **Station**
  Station number for the AdvaSoft station. Station number must be between 1 and 80.
- **ClockMaster**
  Clock master configuration which can be “On” or “Off” (CI525) or “None“, “Master”, “Slave” or “Backup” (CI526). Note that in the case of the CI525 board, only one station on the Advant Fieldbus 100 network can be set up as clock master.
- **RedundantLineSupport**
  Support of redundant lines (two TC625 modems connected). Can be “On” or “Off.”
- **BusMaster**
  Bus master configuration, which can be “On” or “Off.” Normally always set to “On.”

You can modify some of these parameters from the **Configuration** dialog box in the AF100 Communication program.

For an example of a complete AF100 section, see the figure below.

```
...
[AF100Profile]
AdvaSoftStations=11 12
Objects=1500
QuickLoad=On
ConfigTimeStamp=9510121435000
Bus=1
Station=10
BusMaster=On
ClockMaster=None
RedundantLineSupport=Off
EventSize=400
AdvaBuildProjDir=x:\advabuild
Nodes=Contr_1,Contr_2,MMI_1
```

Figure A-2. Example of definitions in the ADVASOFT.INI file

Manually added parameters

- **SupressBoilCheckMessage=On**
  In the case of a stand-alone AdvaSoft station, an error message is written every two minutes to the System Messages log stating, "Application version mismatch." To suppress this message, manually add this parameter. In doing so, you minimize the size of the System Messages log.

- **TimeRecieveStation=station/CPU**
  This parameter forces the real time clock (including date) in a controller, specified by the station/CPU statement. This is done every time a Windows application sends a WM_TIMECHANGE message (for example, the Date/Time application found in the Control Panel). The controller (the CI626 board) must be TimeMaster.

- **ReadLimitsAtStartup=YES**
  This parameter forces AdvaSoft to retrieve all limits for AIS objects from the controller at startup instead of retrieving them from the BOIL file. It reflects all changes made during runtime in the controller. Use this parameter for maximum system safety. The parameter increases the startup time with a factor of:

  timefactor=(Number_of_objects)/2

  (ca 8 SDP calls per second, four limits per object gives
  timefactor=(Number_of_objects)x(4/8))

  Default is NO.

- **NewStations=NO**
  If you press the **Detail** button in the **Network Status** dialog, old versions of AdvaSoft, 1.1/0 and earlier, and controllers stall. To circumvent this problem, enter this parameter when old versions are present in your system. It is also valid for AC110, version 2.0/4 and earlier. Default is YES (only new controllers are present in the system).

- **ShowStationStatusDSP=ON**
  Use only when AC 160 controllers, with multiple CPUs, are present in your system.
  An Advant Controller 160 contains several CPUs and one AF 100 CI board. The status that is written to the Network Status dialog, displayed when you choose the Status item in the AF100 system menu, is the CPU status (not the board status). This is special for the AC 160 because the CPU and the board status are two different entities.

  The status will always be the same for an AC 110, which makes this information redundant.

Default is OFF for this parameter.

```
...
[AF100Profile]
AdvaSoftStations=11 12
Objects=1500
QuickLoad=On
ConfigTimeStamp=9510121435000
Bus=1
Station=10
BusMaster=On
ClockMaster=None
RedundantLineSupport=Off
AdvaBuildProjDir=x:\advabuild
Nodes=Contr_1,Contr_2,MMI_1
SupressBoilCheckMessage=On
TimeRecieveStation=station/CPU
ReadLimitsAtStartup=YES
NewStations=NO
ShowStationStatusDSP=ON
```

Figure A-3. Example on the AF100Profile section in the ADVASOFT.INI file

Appendix B Object Attributes and Commands

B.1 General

The tables below include object attributes that can be subscribed to by the client application.

There is a column for each target object type (e.g., AI type **DAT Based Object** or **Extended DB Elements**) mapped to an AdvaInform object type (e.g., AI or DO) showing how the attribute is supported:

X	the attribute is supported
constant	the attribute has a constant value
<—>	the attribute is not supported.

The attributes supported for different object types are described on the following pages.

For detailed information about the different attributes and commands, refer to the *AdvaInform Object Types Reference Manual.*

You can make cyclic data subscriptions with cycle times down to 100 msec. To get msec update times, set the most significant bit for cycle time parameter to one when calling the User API.

B.2 Analog Input (AI) object type

The following attributes are supported by the Analog Input object type:

Table B-1. Analog Input object type attributes

Attribute	DAT Based Object	Extended DB Elements	Description
NAME	X	X	The name of the object
DESCRIPTION	X	X	The description of the object
VALUE	X	X	The current value of the object(1)
NO_OF_DEC	2	X	Number of decimals
UNIT	X	X	The unit of the object
STATUS	X	X	Represents the status of the object
SUBSYSTEM	X	X	Process section 0-16
CLASS	0	X	Class
RANGE_MIN	X	X	The smallest allowed object value
RANGE_MAX	X	X	The highest allowed object value

Table B-1. Analog Input object type attributes (Continued)

Attribute	DAT Based Object	Extended DB Elements	Description
HI_LIM2	—	X	Alarm limit value HI_LIM2
HI_LIM1	—	X	Alarm limit value HI_LIM1
LO_LIM1	—	X	Alarm limit value LO_LIM1
LO_LIM2	—	X	Alarm limit value LO_LIM2
USE_HI_LIM2	1	X	Alarm limit value HI_LIM2 used
USE_HI_LIM1	1	X	Alarm limit value HI_LIM1 used
USE_LO_LIM1	1	X	Alarm limit value LO_LIM1 used
USE_LO_LIM2	1	X	Alarm limit value LO_LIM2 used
LIM1_TREAT	4	X	Value of event treatment definitions in MV/OS
LIM2_TREAT	4	X	Value of event treatment definitions in MV/OS
ERROR_TREAT	2	X	Value of event treatment definitions in MV/OS

(1) **Scaling of VALUE attribute** (Extended DB Elements only)
The full range defined by the engineering unit is scaled to the I/O board value range.

$$\mathrm{VALUE} = \left(\frac{RANGEMAX - RANGEMIN}{HWVALMAX - HWVALMIN} \times (InputValue - HWVALMIN)\right) + RANGEMIN$$

This yields
- VALUE = RANGEMIN if Input value = HWVALMIN
- VALUE = RANGEMAX if Input value = HWVALMAX

The scaling is reciprocal when *writing* to the VALUE attribute.

Before writing the value (via SDP or DSP) it is checked against the RANGEMIN/RANGEMAX attributes.

The following status bits are supported by the Analog Input object type:

Table B-2. Analog Input object type status bits

Bits in TdStatus	Bit number	DAT Based Object	Extended DB Elements	Description
activate	1	1	X	Object active, i.e., ready to run
error	2	X	X	Indicates the overview status.
updated	3	X	X	Indicates if the object value is updated
UPD_BLK	4	0	X	The value is blocked by ASfW or forced by AdvaBuild
ABOVE_HI_LIM2	9	X	X	Alarm limit transgressed
ABOVE_HI_LIM1	10	X	X	Alarm limit transgressed
BELOW_LO_LIM1	11	X	X	Alarm limit transgressed
BELOW_LO_LIM2	12	X	X	Alarm limit transgressed
ALARM_UNACK	13	X	X	Alarm unacknowledged
DISTURBANCE	15	0	X	The object is in an error state
ALARM_BLK	17	0	X	Alarms are blocked
ALARM_PERIOD_BLK	18	0	X	Value of event set status (EVS_STAT): 0: mode changes do not generate events 1: mode changes do generate events
ERR_CTRL	32	0	X	Value of event set status (EVS_ERR): 0: HW errors do not generate events 1: HW errors do generate events

The following commands are supported by the Analog Input object type:

Table B-3. Commands supported by the AI object type

Command	DAT Based Object	Extended DB Elements	Description
Change value	X	X	Updates the object value(1)
Acknowledge alarm	X	X	Alarm acknowledgment (For information about how this works for old and Extended DB Elements see Section 4.4, Operating Instructions.)
Change H2LIM(2)	0	X	Set the value of H2LIM
Change H1LIM(2)	0	X	Set the value of H1LIM
Change L1LIM(2)	0	X	Set the value of L1LIM
Change L2LIM(2)	0	X	Set the value of L2LIM
Block process update(2)	0	X	Disable updating of VALUE terminal
Unblock process update(2)	0	X	Enable updating of VALUE terminal
Block alarms(2)	0	X	disable event generation
Unblock alarms(2)	0	X	enable event generation

(1) Note that commands to input objects require that the update of the input value is blocked in the target system (Controller).

(2) This command sets the bit in the object status word locally without involving the controller.

B.3 Analog Output (AO) object type

The following attributes are supported by the Analog Output object type:

Table B-4. Attributes of AO object type

Attribute	DAT Based Object	Extended DB Elements	Description
NAME	X	X	The name of the object
DESCRIPTION	X	X	The description of the object
VALUE	X	X	The current value of the object(1)
NO_OF_DEC	2	X	Number of decimals
UNIT	X	X	The unit of the object
STATUS	X	X	Represents the status of the object

Table B-4. Attributes of AO object type (Continued)

Attribute	DAT Based Object	Extended DB Elements	Description
SUBSYSTEM	X	X	Process section 0-16
CLASS	0	X	Class
RANGE_MIN	X	X	The smallest allowed object value
RANGE_MAX	X	X	The highest allowed object value
MAX_LIM	0	X	The highest allowed value given as a command
MIN_LIM	0	X	The lowest allowed value given as a command
ERROR_TREAT	2	X	Value of event treatment definitions in MV/OS
START_VALUE		X	Start value

(1) **Scaling of VALUE attribute** (Extended DB Elements only)
The full range defined by the engineering unit is scaled to the I/O board value range.

$$VALUE = \left(\frac{RANGEMAX - RANGEMIN}{HWVALMAX - HWVALMIN} \times (OutputValue - HWVALMIN)\right) + RANGEMIN$$

This yields
- VALUE = RANGEMIN if Output value = HWVALMIN
- VALUE = RANGEMAX if Output value = HWVALMAX

The scaling is reciprocal when *writing* to the VALUE attribute.

Before writing the value (via SDP or DSP) it is checked against the RANGEMIN/RANGEMAX attributes.

The following status bits are supported by the Analog Output object type:

Table B-5. STATUS bits supported for the AO object type

Bits in Status	Bit number	DAT Based Object	Extended DB Elements	Description
activate	1	1	X	Object active, i.e., ready to run
error	2	X	X	Indicates the overview status
updated	3	X	X	Indicates if the object value is updated
OUTP_BLK	4	0	X	Output to output connection is blocked
MAN_MODE	8	0	X	Object is in manual mode

Table B-5. STATUS bits supported for the AO object type (Continued)

Bits in Status	**Bit number**	**DAT Based Object**	**Extended DB Elements**	**Description**
ON_MAX_LIM	9	0	X	Signals that output value is above high limit
ON_MIN_LIM	12	0	X	Signals that output value is below low limit
ALARM_UNACK	13	0	X	Alarm unacknowledged
ALARM_BLK	17	0	X	Alarms are blocked
ALARM_PERIOD_ BLK	18	0	X	Value of event set status (EVS_STAT): 0: mode changes do not generate events 1: mode changes do generate events

The following commands are supported by the Analog Output object type:

Table B-6. Analog Output object type commands

Command	**DAT Based Objects**	**Extended DB Elements**	**Description**
Change value	X	X	Updates the object value(1)
Set manual mode(2)	0	X	Sets the element to manual mode
Set auto mode(2)	0	X	Sets the element to auto mode
Block alarms	0	X	Disable event generation
Unblock alarms	0	X	Enable event generation

(1) The *Change Value* Command to AO objects requires that the object is in manual mode (use the *Set Manual Mode* command) in the controller.

(2) Manual/Auto mode is set locally without involving the controller.

B.4 Digital Input (DI) object type

The following status bits are supported by the Digital Input object type:

Table B-7. Attributes of DI object type

Attribute	DAT Based Object	Extended DB Elements	MBS	Description
NAME	X	X	X	The name of the object
DESCRIPTION	X	X	X	The description of the object
ERROR_TREAT	2	X	0	Value of event treatment definitions in MV/OS
VALUE_TREAT	2	X	X	Value of event treatment definitions in MV/OS
STATUS	X	X	X	Represents the status of the object
SUBSYSTEM	X	X	X	Process section 0-16
CLASS	0	X	X	Class

The following attributes are supported by the Digital Input object type:

Table B-8. STATUS bits supported for the DI object type

Bits in Status	Bit number	DAT Based Object	Extended DB Elements	MBS	Description
activate	1	1	X	1	Object active, i.e., ready to run
error	2	X	X	X	Indicates the overview status
updated	3	X	X	X	Indicates if the object value is updated
UPD_BLK	4	0	X	0	The value is blocked by ASfW or forced by AdvaBuild
ERR_CTRL	8	0	X	0	Value of event set status (EVS_ERR): 0:HW errors do not generate events 1:HW errors do generate events
VALUE	9	X	X	X	Current value
NORM_POSN	11	X	X	X	Normal position
ALARM_UNACK	13	X	X	0	Alarm unacknowledged

Table B-8. STATUS bits supported for the DI object type (Continued)

Bits in Status	**Bit number**	**DAT Based Object**	**Extended DB Elements**	**MBS**	**Description**
DISTURBANCE	15	X	X	0	The object is in an error state
ALARM_BLK	17	0	X	0	Alarms are blocked
ALARM_PERIOD_ BLK	18	0	X	0	Value of event set status (EVS_STAT): 0: mode changes do not generate events 1: mode changes do generate events

The following commands are supported by the Digital Input object type:

Table B-9. Commands supported by the DI object type

Command	**DAT Based Object**	**Extended DB Elements**	**MBS**	**Description**
Set value On	X	X	—	Sets the object value to 1(1)
Set value Off	X	X	—	Sets the object value to 0(1)
Acknowledge alarm	X	X	—	Alarm acknowledgment (For information about how this works for old and Extended DB Elements, see Section 4.4, Operating Instructions.)
Block process update(2)	0	X	—	Disable updating of VALUE terminal
Unblock process update(2)	0	X	—	Enable updating of VALUE terminal
Block alarms(2)	0	X	—	Disable event generation
Unblock alarms(2)	0	X	—	Enable event generation

(1) Note that commands to input objects require that the update of the input value be blocked in the target system (Controller).

(2) This command sets the bit in the object status word locally without involving the controller.

B.5 Digital Output (DO) object type

The following attributes are supported by the Digital Output object type:

Table B-10. Digital Output object type attributes

Attribute	DAT Based Object	Extended DB Elements	Description
NAME	X	X	The name of the object
DESCRIPTION	X	X	The description of the object
ERROR_TREAT	2	X	Value of event treatment definitions in MV/OS
STATUS	X	X	Represents the status of the object
SUBSYSTEM	X	X	Process section 0-16
CLASS	0	X	Class

The following status bits are supported by the Digital Output object type:

Table B-11. STATUS bits supported for the DO object type

Bits in Status	Bit number	DAT Based Object	Extended DB Elements	Description
activate	1	1	X	Object active, i.e., ready to run
error	2	X	X	Indicates the overview status
updated	3	X	X	Indicates if the object value is updated
OUTP_BLK	4	0	X	Output to output connection is blocked
MAN_MODE	8	1	X	Object is in manual mode
VALUE	9	X	X	Current value
START_VALUE	11	X	X	Start value
ALARM_UNACK	13	0	X	There is an unacknowledged alarm pending

Table B-11. STATUS bits supported for the DO object type (Continued)

Bits in Status	**Bit number**	**DAT Based Object**	**Extended DB Elements**	**Description**
ALARM_BLK	17	0	X	Alarms are blocked (always TRUE)
ALARM_PERIOD_BLK	18	0	X	Value of event set status (EVS_STAT): 0: mode changes do not generate events 1: mode changes do generate events

The following commands are supported by the Digital Output object type:

Table B-12. Commands supported by the DO object type

Command	**DAT Based Object**	**Extended DB Elements**	**Description**
Set value On	X	X	Sets the object value to 1(1)
Set value Off	X	X	Sets the object value to 0(1)
Set manual mode(2)	—	X	Sets the element to manual mode
Set auto mode(2)	—	X	Sets the element to auto mode
Block alarms(3)	—	X	Disable event generation
Unblock alarms(3)	—	X	Enable event generation

(1) The *Change Value* Command to DO objects requires that the object be in manual mode (use the *Set Manual Mode* command) in the controller.
(2) Only set locally without involving controller.
(3) This command sets the bit in the object status word locally without involving the controller.

B.6 Memory block elements

This object is mapped towards the DAT object.

B.7 DAT object type

The following attributes are supported by the DAT object type:

Table B-13. DAT object type attributes

Attribute	Description
NAME	The name of the object
VALID	Represents the status of the object
VAL_TYPE	Value type
B0_VAL	Boolean value 0
B1_VAL	Boolean value 1
B2_VAL	Boolean value 2
. .	
B31_VAL	Boolean value 31
IW_VAL	Integer word value
IL_VAL	Integer long value
R_VAL	Real value

The following commands are supported by the DAT object type:

Table B-14. DAT object type commands

Type of function	Description
Change first bit in Boolean value	Sets the first bit (attribute B0_VAL) of the object value to 1 or 0. This command overwrites the remaining 31 bits (attributes B1_VAL to B31_VAL) of this DAT object.
Change single bit in Boolean value	Sets a single bit of the object value to 1 or 0. This command is only available if the object type in the controller is a MB.
Change integer word value	Updates the object value. This command overwrites the upper 2 bytes of this object type.
Change integer long value	Updates the object value.
Change integer real value	Updates the object value.

B.8 System Status object type attributes

The following attributes are supported by the System Status object type:

Table B-15. System Status object type attributes

Attribute	Description
NAME	The name of the object (“SYSTEM_STATUS”)
NODE	Station number of own node
STATUS	Status of own node
NETWS	Network status, of the type `OmfNodeNetwArray`, see tables below

The following status bits are supported by the System Status object type:

Table B-16. System Status object type status bits

Bits in NodeStatus	Bit number	Description
IMPLEMENTED	1	Is set if status is implemented (always set)
STATUS_VALID	2	Is set if status is valid (always set)
ERROR	3	Is set for an error condition

The `OmfNodeNetwArray` contains the following fields:

Table B-17. OmfNodeNetwArray object fields

Attribute	Description
max_is	Always 180
last_is	The index of the last used entry in `array`
array	Network status, an array containing 180 elements of type `OmfNodeNetw`. Only the array indices 1…80 are used.

The `OmfNodeNetw` contains the following fields:

Table B-18. `OmfNodeNetw` *object fields*

Fields in `Netwtype`	**Description**
TYPE	Always set to -1 (=unknown)
NETW	Own network number
NODE	Station address of node
STATUS	Network status (type `NetwStatus`)

B.9 Time Tagged Event (EACOLLECT) object type

The EACOLLECT object type supports no subscription of specific attributes. Use "TT_Event" as the object name in the event request. You do not have to configure the TT_EVENT object--it is defined implicitly. There is only one TT-Object in the AF100 Interface.

The following event information is returned in the event data:

Table B-19. TT_Event information

Attribute	**Description**
objectId	In this structure only the field `sequence` is defined: `sequence` = index of the object which sent the event
objectName	The name of the object which sent the event
objectDescription	The description of the object which sent the event
eventID	The event identifier (always 0)
eventParameters	The event parameters (see Table B-20)

The `eventParameters` attribute consists of the following fields:

Table B-20. Structure of `eventParameters` attribute

Field	Description
eventReason	Describes what kind of event has occurred
eventAttribute	Describes which attribute the event occurred on
objectEngUnit	The unit of the object (copy of the object's UNIT attribute)
processSection	The process section to which the object belongs (copy of the object's SUBSYSTEM attribute)
valueType	Type of value that is passed (DCX_REALV|DCX_INTV|DCX_DIGV)
valueFloat	Floating point value. Only used if `valueType` = `DCX_REALV`
valueInteger	Integer value. Only used if `valueType` = `DCX_INTV`
valueBoolean	Boolean value. Only used if `valueType` = `DCX_DIGV`
errTreatReference	Event Treatment pointer. Describes how to treat the event. This parameter is a static object attribute (taken from the BOIL file).
timeQuality	0 = correct time 1 = no time 2 = uncertain time
eventTime	The event time is the number of seconds since 00:00:00, January 1, 1970
eventMillitm	Fractional time in microseconds
blockConditions	Always 0

The AF100 Interface produces the following events (rows shaded in grey indicate that this event applies only to extended DB Elements).

Table B-21. Generated Events

eventReason	eventAttribute	eventTreatment	Description
DCX_EVENT_ON	DCX_EVENT_LOST	EVENT_TREAT1(1)	Lost events in controller (AC110)
DCX_ALARM_ON/DCX_ALARM_OFF	DCX_EERROR	ERR_TR from BOIL file	AI/AO/DI/DO object reports hardware error on input/output module
DCX_BLOCKED	DCX_EPROC_UPDATE	StatusEventTreat from ADVASOFT.INI	AI/DI object reports disabling of process update (command `Block process update`)
DCX_DEBLOCKED	DCX_EPROC_UPDATE	StatusEventTreat from ADVASOFT.INI	AI/DI object reports enabling of process update (command `Unblock process update`)
DCX_EVENT_ON	DCX_EAUTO	StatusEventTreat from ADVASOFT.INI	AO/DO object reports setting to auto mode (command `Set auto mode`)
DCX_EVENT_ON	DCX_EMANUAL	StatusEventTreat from ADVASOFT.INI	AO/DO object reports setting to manual mode (command `Set manual mode`)
DCX_BLOCKED	DCX_EDIST_ALARM	StatusEventTreat from ADVASOFT.INI	AI/AO/DI/DO object reports alarms blocked
DCX_DEBLOCKED	DCX_EDIST_ALARM	StatusEventTreat from ADVASOFT.INI	AI/AO/DI/DO object reports alarms unblocked
DCX_ACK_LIST	DCX_EDUM_EV_PROP	ERR_TR from BOIL file	AI/AO/DI/DO object reports alarm acknowledgment
DCX_VALUE_CHANGE	DCX_EIND_VALUE	ERR_TR from BOIL file	One or more bits of an MB object changed their value
DCX_ALARM_ON	DCX_EIND_VALUE	NORM_TR from BOIL file (1)	DI objects value change to abnormal value
DCX_ALARM_OFF	DCX_EIND_VALUE	NORM_TR from BOIL file (1)	DI objects value change to normal value
DCX_EVENT_ON	DCX_EOUTPUT	StatusEventTreat from ADVASOFT.INI	DO objects value change to abnormal value
DCX_EVENT_OFF	DCX_EOUTPUT	StatusEventTreat from ADVASOFT.INI	DO objects value change to normal value

Table B-21. Generated Events (Continued)

eventReason	eventAttribute	eventTreatment	Description
DCX_ALARM_ON	DCX_EHI_LIM2 DCX_EHI_LIM1 DCX_ELO_LIM1 DCX_ELO_LIM2	LIM_1_TR(2), LIM_2_TR(3) from BOIL file	AI object transgressed alarm limit when going in the normal to abnormal direction
DCX_ALARM_OFF	DCX_EHI_LIM2 DCX_EHI_LIM1 DCX_ELO_LIM1 DCX_ELO_LIM2	LIM_1_TR(2), LIM_2_TR(3) from BOIL file	AI object transgressed alarm limit when going in the abnormal to normal direction
DCX_VAL_CHANGE	DCX_EHI_LIM2 DCX_EHI_LIM1 DCX_ELO_LIM1 DCX_ELO_LIM2	StatusEventTreat from ADVASOFT.INI	AI object's alarm limit changed by operator

(1) For DAT based objects; the eventTreatment depends on the configuration made in the AF100 Object Configurator. If you check the **Use normal value** checkbox for DI/DO objects, in the configuration dialog box, the eventTreatment is ALARM_TREAT1 (2). If not checked, the eventTreatment is EVENT_TREAT1 (1).

(2) (for DCX_EHI_LIM1, DCX_ELO_LIM1)

(3) (for DCX_EHI_LIM2, DCX_ELO_LIM2)

Appendix C COIL file structures

C.1 General

There are three COIL files, COIL0000.C00, COILESET.B00 and COILSERV.B00. All three are in ASCII format. They contain configuration information for the AF100 interface.

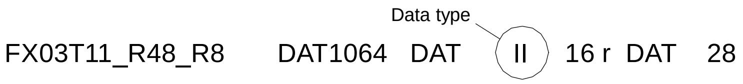
The COIL files are generated by the BCB utility included in the AdvaBuild product. Normally, they are never to be modified by the user with one exception: when the integer word DAT object type is used. This requires some user modifications in the COIL0000.C00 file. The COILSET and the COILSERV files are not affected by this modification. The DAT object type is only supported by the APC, not by the AC110.

DAT object related changes

When the DAT object type is used, you must make the following additions and changes in each DatLine in the COIL0000.C00 file:

- Define the integer word DAT object type. Do this by setting the data type for each Datline to II (DatLine; column position 41).

  See Section C.2.6, DatLine for more information.

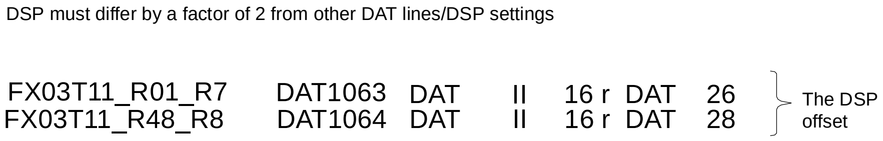


```
FX03T11_R48_R8      DAT1064   DAT    II   16 r  DAT    28
```

Figure C-1. Definition of data type

- Ensure the DSP offset (DatLine; column position 59) for each DAT line differs by a factor of 2 between each Datline, starting with 0 (for standard DATs, the factor is 4).

  See Section C.2.6, DatLine for more information.


DSP must differ by a factor of 2 from other DAT lines/DSP settings
The DSP offset

```
FX03T11_R01_R7     DAT1063  DAT    II    16 r  DAT   26
FX03T11_R48_R8     DAT1064  DAT    II    16 r  DAT   28
```

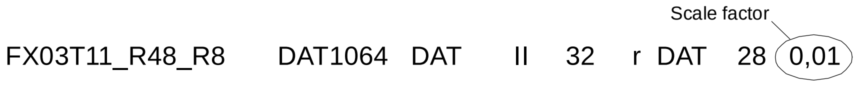
Figure C-2. Definition of DSP offset

- Add a scaling factor for each DAT line (DatLine; column position 71).

  Available scaling factors are 0.001, 0.01, 1, 10, 100 and 1000.

NOTE

Use a period (.) as item separator in the scale factor.



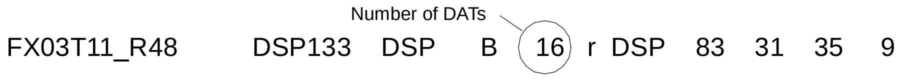
```
FX03T11_R48_R8      DAT1064   DAT      II    32    r  DAT    28   0,01
```

Figure C-3. Definition of scaling factor

See Section C.2.6, DatLine for more information.

DSP object changes

- You can set the number of DATs in the DSP line (DataSetLine; column position 44) up to 16.



```
FX03T11_R48         DSP133    DSP     B    16   r  DSP    83    31    35    9
```

Figure C-4. Insertion point for the number of DATs

See Section C.2.5, DataSetLine for more information.

C.2 COIL0000.C00 file

C.2.1 Structure

The COIL file is constructed of a standard set of line types: StartLine, HeadLine, TerminalLine, DataSetLine, DatLine and EndLine. Each line type has a specific place in the COIL file. Below is an example of what the structure of the COIL0000.C00 file looks like:

Table C-1. Structure of the COIL0000.COO file

```
StartLine
HeadLine
TerminalLine MASTER terminal
TerminalLine TIMESYNC terminal
TerminalLine CABLE terminal
DataSetLine      }
DatLine 1        }
.                }   Repeated for
.                }   each set of data.
DatLine 8        }
DataSetLine
DatLine 1
.
.
DatLine 8
EndLine
```

Each line in turn has its own standardized organization, which is explained in the following sections.

Below is an example of an actual COIL0000.C00 file:

```
COIL0000.C00 file                                          Comments

COIL0000.C00  1.5/0 951011133318000 STA                    StartLine
CI1       CI1    0,  0,  0,212  HEA    ADVASOFTPC16        HeadLine
CI1       CI1    MASTER    B   1   rw AF1     1            TerminalLine
CI1       CI1    TIMESYNC  A       rw AF1    NONE          TerminalLine
CI1       CI1    CABLE     I   16     rw AF1     1         TerminalLine
F52T10_BL1       DSP1   DSP    B   8   r   DSP    1 2 1 9  DataSetLine
F52T10_BL1_B1    DAT1   DAT    B   32  r   DAT    0        DatLine 1
. .
. .
F52T10_BL1_R2    DAT8   DAT    R    32     r  DAT 28       DatLine 8
1F52_1           DSP2   DSP    B     8     r  DSP 2229     DataSetLine
1F52_1_B1        DAT9   DAT    B    32     r  DAT 0        DatLine 9
. .
. .
1F52_1_R2        DAT16  DAT    R   32      r  DAT28        DatLine 16
DSP3             DSP3   DSP    B     8     r  DSP 3239     DataSetLine
DSP3.IL1         DAT17  DAT    IL   32     r  DAT     0    DatLine 17
. .
. .
DSP3.IL8         DAT24  DAT    IL  32      r  DAT28        DatLine 24
F52T11_B01       DSP4   DSP    B   8   r   DSP 4 2 5 9     DataSetLine
. .
. .
FX03T11_R48      DSP133 DSP    B   8   r   DSP 83 31 35 9  DataSetLine
FX03T11_R48_R1   DAT1057 DAT   R    32 r   DAT     0       DatLine 1057
. .
. .
FX03T11_R48_R8   DAT1064 DAT   R    32 r   DAT 28          DatLine 1064
COIL0000.C00          END                                  EndLine
```

Figure C-5. Example of a COIL0000.C00 file

The content of the COIL0000.C00 file above has been edited and marked with dots so it will be easier to understand. When you edit the file, use a text font with fixed character spacing to achieve a text column layout.

C.2.2 StartLine

The StartLine consists of the file name, the version, a timestamp and the access STA. The timestamp is generated by the BCB.

Example (in compressed form):

```
COIL0000.C00 1.4/0  940817122837190                STA
```

Detailed description of StartLine

Table C-2. Content of the StartLine

Column From	To	Description	Limits
01	12	Filename	Alphanumeric
13		blank	
14	15	Major part of version	1 to 99
16		"."	
17		Minor part of version	0 to 9
18		"/"	
19	22	Revision part of version	Alphanumeric
23		Blank	
24	25	Year	0 to 99
26	27	Month	1 to 12
28	29	Day	1 to 31
30	31	Hour	0 to 23
32	33	Minute	0 to 59
34	35	Second	0 to 59
36	38	Millisecond	0 to 999
39	54	Blank	
55	57	Access STA	Alphanumeric

C.2.3 HeadLine

The HeadLine consists of the object name, the item designation of the processor module, the controller address, a node type code, the access HEA, the project name and the target name.

Example (in compressed form):

```
CI1           CI1      0,  0,  0,10     2  HEA SURSELVA  WATER
```

Detailed description of the HeadLine

Table C-3. Content of the HeadLine

Column From	To	Description	Limits
01	20	Object name	Alphanumeric
21		Blank	
22	30	Item designation	Alphanumeric
31		Blank	
32	34	Net No. of controller station	0.255
35		","	
36	38	Node No. of controller station	0 to 255
39		","	
40	42	Bus No. of controller station	0 to 255
43		","	
44	45	Station No. of controller station	0 to 79
46		Blank	
47	51	Node type code, 1 = AC110 2 = AdvaSoft for Windows	1, 2
52	54	Blank	
55	57	Access HEA	Alphanumeric
58		Blank	
59	66	Project name	Alphanumeric
67		Blank	
68	75	Target name	Alphanumeric
76		Blank	

C.2.4 TerminalLine

Three types of TerminalLines exist in the COIL0000.C00 file: MASTER, TIMESYNC and CABLE.

Example (in compressed form):

```
CI1            CI1      MASTER      B       1    rw    AF1     1
```

Detailed description of TerminalLine

Table C-4. Content in TerminalLine

Column From	To	Description	Limits
01	20	Object name	Alphanumeric
21		Blank	
22	30	Item designation	Alphanumeric
31		Blank	
32	39	Element description. One of these: MASTER TIMESYNC CABLE	Alphanumeric
40		Blank	
41	42	Data type	A, B, I, IL or R
43		Blank	
44	46	Data length in bits	1 to 32
47		Blank	
48	49	Bit number	0 to 32 or blanks
50	51	Blank	
52	53	Access direction	r, w or rw
54		Blank	
55	57	Access	AF1

Table C-4. Content in TerminalLine (Continued)

Column From	To	Description	Limits
58		Blank	
59	64	For Terminal MASTER: 0 = SLAVE mode 1 = MASTER mode TIMESYNC: NONE MASTER CABLE: 1 = One cable used 2 = two physical cables are needed for redundancy	Alphanumeric

C.2.5 DataSetLine

Each set of data has it own section in the COIL0000.C00 file. Each section starts with a DataSetLine.

Example (in compressed form):

```
DSP1         DSP1     DSP     B     8    w   DSP   2   2   2  12
```

Detailed description of DataSetLine

Table C-5. Contents of the DataSetLine

Column From	To	Description	Limits
01	20	Object name	Alphanumeric
21		Blank	
22	30	Item designation	Alphanumeric
31		Blank	
32	39	Element description DS for the DB element Data Set DSP for the DB element Data Set Peripheral	Alphanumeric
40		Blank	
41	42	Blank	
43		Blank	
44	46	Number of DatLines	0 to 8
47		Blank	
48	49	Bit Number	0 to 32 or blank
50	51	Blank	
52	53	Access direction	"r", "w" or "rw"
54		Blank	
55	57	Access DS or DSP	Alphanumeric
59	63	Ordinal DS number in COIO	1 to 65 535
64		Blank	
65	69	Value of terminal STATION for DB element DS and access direction "w" STATION = Station No. of controller station	1 to 80

Table C-5. Contents of the DataSetLine (Continued)

Column From	**To**	**Description**	**Limits**
70			
71	75	Value of terminal IDENT	1 to 50
77	81	Value of terminal CYCLETIM (scan_ftr) (2**CYCLETIM= x ms, i.e., CYCLETIM=9 --> 512 ms)	0 to 12

C.2.6 DatLine

Example (in compressed form:

```
ARVE         DAT4      DAT       R    32      W    DAT        4
```

Detailed description of the DatLine

Table C-6. Content of the DatLine

Column From	To	Description	Limits
01	20	Object name	Alphanumeric
21		Blank	
22	30	Item designation	Alphanumeric
31		blank	
32	39	Element description DAT for the DB element DAT (B, I, II, IL, R)	Alphanumeric
40		Blank	
41	42	Data type	B, I, II, IL, R
43		Blank	
44	46	Data length in bits	1 to 32
47		Blank	
48	49	Bit number	0 to 32 or blanks
50	51	Blank	
52	53	Access direction	“r”, “w” or “rw”
54		Blank	
55	57	Access DAT	Alphanumeric
59	63	Offset in DS or DSP	0 to 32
64		Blank	
65	70	Blank	
71	75	Scaling of DAT values	

C.2.7 EndLine

The COIL0000.C00 file ends with an EndLine.

Example (in compressed form):

```
COIL0000.c000                                                END
```

Detailed description of the EndLine

Table C-7. Content of the EndLine

Column From	To	Description	Limits
01	12	Filename	Alphanumeric
13	54	Blank	
55	57	Access END	Alphanumeric

INDEX

A

About... 2-9, 4-11
AdvaCommand for Windows 1-3
AdvaInform
  DDE server 1-3
  User API 1-3
Advant Controller 1-6
Advant Fieldbus 100
  Interface 1-1, 1-9
Advant Station 1-6
ADVASOFT.INI 2-3, 3-1, A-1
AdvaSoftStations A-1
AF100
  Communication software 1-9, 1-12
  Object configuration software 1-9
AF100 Communication software
  Commands 1-13
  Data subscription 1-13
  Diagnostics 1-15
  Event handling 1-13
  Hardware configuration 1-15
AF100 Object Configuration software 3-48
  Configure object attributes 1-16
  Connection to event channels 1-16
  Setup of I/O object type 1-16
  Setup of object name 1-16
AF100SYM.BAK 3-49
AF100SYM.DTA 3-49
AI 1-14, B-1
Analog Input B-1
Analog Output B-4
AO 1-14, B-4
APB 1-6
APC 1-6
Application software 1-3

B

BCB 1-6, 3-1
BOIL files
  Backup 5-1
  BOIL file 1-6
  BOIL0000.C00 3-19
  Creating 3-18
  distributing 3-19
BUS 3-44
Bus administration enabled 4-7
Bus Configuration Builder 3-1
Bus Master 1-7, A-3
Bus, Station 3-40, 3-43

C

Capacity 3-2
CI board 2-4
CI board Configuration
  AF 100 Parameters
    Bus administration 4-5
    Redundant line supported 4-5
    Time synchronization 4-5
  Network connection
    Connect 4-4
    Disconnect 4-4
CI board Errors
  Invalid telegram 5-2
  Invalid telegram data 5-2
  Multiple source 5-2
  Permanent sender 5-2
CI board installation 2-3
CI board Start-up 4-1
CI board status information 4-6
Client application 1-7, 4-3
Clock Master 1-7, A-3
COIL 1-6
COIL0000.C00 3-42
COILESET.B00 3-42
COILSERV.B00 3-42
Communication interface board 1-9, 2-2
Config... 4-12
ConfigTimeStamp A-2
Configuration data files 3-1
Configuration of AI/AO objects 3-50
Configuration of DI/DO objects 3-50
Configuration of the CI525 4-4
Connection cable 2-4
Cyclic subscriptions 3-39

D

DAT 1-14, 3-42, B-11
DAT Based Objects 1-7
DAT object B-11
DataSet Peripheral 1-6, 3-38, 3-46
DBDATA directory 3-44
Default station number 2-5
Device Id 4-7
Device state 4-7

DI 1-14, B-7
Digital Input B-7
Digital Output B-9
DO 1-14, B-9
DSP 1-6
Dynamic data 4-7

E

Electro-static discharge 2-2
Emergency shutdown 2-8
Equipment requirements 1-3
Error information 4-8
Error messages
    Hardware Errors 5-1
    Software Errors 5-1
ESD 2-2
Event Set 1-6, 1-13, 3-42, 3-46
EVS 1-6, 1-13, 3-42
Extended DB Elements 1-7

F

Fatal device errors 5-2
FCE 1-6, 3-1, 3-41, 3-43 to 3-44
Function chart 1-7
Function Chart Editor 1-6, 3-1

G

Grounding 2-1

I

I/O address 2-3
IoBaseAddress
    A-2
IRQ level 2-2
IRQlevel
    A-2
ISA bus 1-7

L

LED 2-9, 5-1
Line 1 active 4-7

M

modem 2-4

N

Network status 4-8
Normal shutdown 2-8

O

Object 1-8
Object configuration 3-46
Object Configuration utility
    Engineering unit 3-47
    Normal value 3-48
    Object description 3-47
    Object name 3-47
    Object type 3-47
    Process section 3-47
    Value ranges 3-47
Object Groups 1-10
    DAT based Objects 1-10
        test configuration 3-21
    Extended DB Elements 1-10
        test configuration 3-4
Objects
    A-1

P

PC 1-7
PC element 1-8
PC program 1-8
Performance 3-2
Power 2-9
Process errors 5-2
Product verification 2-9
Project name 3-40, 3-43
Project Target 3-40, 3-43

R

Receiving DataSet Peripheral 3-38
Reconfiguration 4-7
Redundant line 4-6
RedundantLineSupport A-3
Resolve 4-11
Runtime 4-2

S

SDP 1-7
Sending data 3-38
Service Data Protocol 3-38
Shutdown 4-2
Signal config 4-7
Start-up procedures 2-8
Station A-3
Station number 2-8, 4-7
Status... 2-9, 4-12
System errors 5-2

System Status object B-12
SYSTEM_STATUS 1-14

T
Target type 3-40, 3-43
TC625/TC630 modem 1-9, 2-1
Terminations 2-4
Time synchronization 4-6
Time Tagged Event B-13
TT_EVENT 1-14, B-13
TT_Event 1-7

U
User interface 1-17

3BSE 006 238R0201

July, 1997